Α.Τ.Ε.Ι. Καλαμάτας παράρτημα Σπάρτης
Τμήμα: Τεχνολογία Πληροφορικής
και Τηλεπικοινωνιών

- Πτυχιακή Εργασία -

«Ανάπτυξη πολυνηματικής (multi-threaded) εφαρμογής διακομιστή-πελάτη (Client-Server) σε C#»

Φοιτητές:
Τσάβος Σωτήρης Α.Μ : 2007105
E-mail: ProKressIV@hotmail.com
Χατζής Νικήτας Α.Μ : 2007254
E-mail: Bonge19@hotmail.com
Εισηγητής:
Καλαντζή Λαμπρινή

Απρίλιος 2013

Πνευματικά Δικαιώματα

Copyrights © Τσάβος Σωτήριος και Χατζής Νικήτας, 2012
Με την επιφύλαξη παντός δικαιώματος. All rights reserved.

Υπεύθυνη Δήλωση

- Βεβαιώνουμε ότι είμαστε συγγραφείς αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχαμε για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία.

- Επίσης έχουμε αναφέρει τις όποιες πηγές από τις οποίες κάναμε χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες.

- Επίσης βεβαιώνουμε ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμάς προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Τεχνολογίας Πληροφορικής και Τηλεπικοινωνιών του Α.Τ.Ε.Ι. Καλαμάτας παραρτήματος Σπάρτης.

Ευχαριστίες

Θα θέλαμε να ευχαριστήσουμε την κ. Λαμπρινή Καλαντζή για την πολύτιμη βοήθεια και καθοδήγηση της σε διάφορα θέματα που είχαν προκύψει με την πτυχιακή μας εργασία.

Θα θέλαμε ακόμα να ευχαριστήσουμε θερμά τον κ. Παναγιώτη Τριτάκη κυρίως για την εμπιστοσύνη που μας έδειξε και την υπομονή που έκανε κατά τη διάρκεια της υλοποίησης της πτυχιακής εργασίας. Όπως επίσης και για την πολύτιμη βοήθεια και καθοδήγηση του κατά την επίλυση διάφορων θεμάτων. Να θυμίσουμε πως ήταν ο κύριος βοηθός μας, μιας και με αυτόν δουλέψαμε το μεγαλύτερο διάστημα και ήταν ο άμεσος αποδέκτης των ερωτημάτων που δημιουργήθηκαν κατά την διάρκεια της πτυχιακής.

Τέλος θα θέλαμε να απευθύνουμε τις ευχαριστίες μας στους γονείς μας, οι οποίοι στήριξαν τις σπουδές μας με διάφορους τρόπους, φροντίζοντας για την καλύτερη δυνατή μόρφωση μας.

Πρόλογος

Η συγκεκριμένη εργασία αφορά στην υλοποίηση μιας εφαρμογής διαδικτυακού παιχνιδιού βασισμένη στους κανόνες ενός «διάσημου» παιχνιδιού όπως το Πόκερ. Στην εφαρμογή που αναπτύχθηκε οι χρήστες θα μπορούν να συνδέονται χρησιμοποιώντας τα προσωπικά στοιχεία σύνδεσης και θα αποκτούν το δικαίωμα να συμμετάσχουν σε διάφορες παρτίδες πόκερ, να συνομιλούν με τους υπόλοιπους συνδεδεμένους χρήστες και να παρακολουθούν παιχνίδια άλλων χρηστών. Η εφαρμογή αναπτύχτηκε για την εκπόνηση της πτυχιακής εργασίας μας στο τμήμα Τεχνολογίας Πληροφορικής και Τηλεπικοινωνιών.

Η εφαρμογή η όποια σχεδιάστηκε στα πλαίσια της πτυχιακής εργασίας, έγινε με πρωταρχικό σκοπό την επαφή και εμβάθυνση με μια από τις πλέον σύγχρονες και διαδεδομένες γλώσσες προγραμματισμού (C#), καθώς και την εξοικείωση μας με την αρχιτεκτονική εξυπηρετητή - πελάτη. Στα πλαίσια της πτυχιακής εργασίας μας δόθηκε η ευκαιρία της χρήσης του καινοτόμου εργαλείου "Microsoft XNA" για την υλοποίηση της εφαρμογής. Επίσης λόγω του μεγέθους της εργασίας γνωρίσαμε μια διαφορετική προγραμματιστική σκέψη και οπτική γωνία με την οποία έπρεπε να βλέπουμε το «πρόβλημα» για να φτάσουμε στην γρηγορότερη και αποτελεσματικότερη επίλυση του. Βασικό μας μέλημα καθ' όλη την διάρκεια αυτής της εργασίας και βασικός μας στόχος ήταν η απόκτηση εμπειριών και γνώσεων πάνω σε έναν ελκυστικό και ενδιαφέροντα τομέα όπως είναι ο διαδικτυακός προγραμματισμός.

Το συγκεκριμένο έγγραφο έχει ως σκοπό, αρχικά να εισάγει τον αναγνώστη στην ιδέα της συγκεκριμένης εργασίας, όπως επίσης και στις τεχνολογίες, στις μεθόδους και στις τεχνικές που αξιοποιήθηκαν για την ολοκλήρωση της εργασίας. Επιπλέον, στόχος του κειμένου είναι όσο το δυνατό πληρέστερη τεκμηρίωση της εφαρμογής που αναπτύχθηκε.

Έτσι λοιπόν το πρώτο κεφάλαιο του συγγράμματος περιέχει αρχικά μια εισαγωγή στην γενική ιδέα της εφαρμογής που υλοποιήθηκε. Έπειτα γίνεται μία αναφορά στην γλώσσα προγραμματισμού που χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής. Υπάρχει ακόμα αναφορά στην τεχνολογία των βάσεων δεδομένων που χρησιμοποιήσαμε. Επίσης έχουμε μια εκτενή περιγραφή της βασικής τεχνολογίας του "Microsoft XNA". Τέλος έχουμε την παρουσίαση δυο εργαλείων της γλώσσας προγραμματισμού της C#, του εργαλείου που χρησιμοποιήθηκε για την δημιουργία των γραφικών στην πλευρά της εφαρμογής του πελάτη και του εργαλείου που χρησιμοποιήθηκε για την μεταφορά των μηνυμάτων μεταξύ εξυπηρετητών και πελατών.

Το επόμενο κεφάλαιο περιγράφει τις απαιτήσεις της εφαρμογής που υλοποιήθηκε όπως αυτές καθορίζονται από την θεωρία ανάπτυξης λογισμικού. Αρχικά υπάρχει μια ανάλυση της αρχιτεκτονικής εξυπηρετητή - πελάτη όπως αυτή εφαρμόζεται στο σύστημα που αναπτύξαμε, ενώ έπειτα καταγράφονται οι λειτουργικές απαιτήσεις των εξυπηρετητών. Στην ενότητα που ακολουθεί γίνεται προσδιορισμός των λειτουργικών απαιτήσεων της εφαρμογής πελάτη και τέλος παρουσιάζεται η ανάλυση των μη-λειτουργικών απαιτήσεων της εφαρμογής.

Στο τρίτο κεφάλαιο έχουμε την καταγραφή των στατικών διαγραμμάτων της UML και συγκεκριμένα του διαγράμματος κλάσεων και συσχετίσεων αυτών. Αρχικά στην πρώτη ενότητα του κεφαλαίου γίνεται ανάλυση των κλάσεων του εξυπηρετητή

κατάστασης. Έπειτα παρατίθεται μία περιγραφή των κλάσεων του κεντρικού εξυπηρετητή, ενώ στη συνέχεια προσδιορίζονται οι κλάσεις που αφορούν στον εξυπηρετητή παιχνιδιού και γίνεται παρουσίαση των κλάσεων της εφαρμογής από την πλευρά του πελάτη.

Το τέταρτο και τελευταίο κεφάλαιο περιλαμβάνει την ανάλυση των δυναμικών διαγραμμάτων του συστήματος. Συγκεκριμένα στην πρώτη ενότητα του κεφαλαίου έχουμε την ανάλυση των διαγραμμάτων περιπτώσεων και των συσχετίσεων τους. Στην επόμενη ενότητα γίνεται ανάλυση των διαγραμμάτων δραστηριοτήτων του συστήματος, ενώ στην τελευταία ενότητα αναλύονται τα διαγράμματα ακολουθίας του συστήματος και των μηνυμάτων που ανταλλάσσονται μεταξύ των στοιχείων αυτού.

-Κεφάλαιο 1°-

Εισαγωγή

1. Εισαγωγή

Το κεφάλαιο αυτό περιγράφει την ιδέα στην όποια βασίστηκε η εφαρμογή που υλοποιήθηκε καθώς και τα εργαλεία και τις τεχνολογίες που συνδυάστηκαν μέσα στον κώδικα, με βασικό σκοπό την επίτευξη του επιθυμητού αποτελέσματος. Βασικό κριτήριο επιλογής των εργαλείων ήταν το να είναι εργαλεία ή τεχνολογίες με τα οποία δεν είχαμε έρθει σε επαφή στο παρελθόν. Παράλληλα θέλαμε να έχουμε πρόσβαση στον κώδικα τους ώστε να μπορούμε να εμβαθύνουμε στον τρόπο λειτουργίας τους, αλλά και για να γίνεται δυνατή η παραμετροποίηση και η περαιτέρω εξέλιξη και βελτίωση του κώδικα όπου αυτό κρίνονταν αναγκαίο. Τέλος εστιάσαμε στον πειραματισμό μας πάνω στον υπάρχοντα κώδικα για την βελτιστοποίηση της εφαρμογής.

Το συγκεκριμένο κεφάλαιο απευθύνεται στους αναγνώστες που δεν είναι εξοικειωμένοι με το θέμα της εργασίας και με τις τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίησή της. Οι αναγνώστες που γνωρίζουν τα θέματα των παρακάτω ενοτήτων μπορούν να παραλείψουν αυτές τις ενότητες ή όλο το κεφάλαιο και να μεταβούν στο επόμενο.

1.1 Γενική Ιδέα της Εφαρμογής

Το πόκερ είναι ένα από τα πιο διάσημα παιχνίδια με τράπουλα της εποχής μας με πάρα πολλούς θαυμαστές και παίκτες παγκοσμίως. Μπορούν να παίξουν από 2 έως 10 παίκτες σε ένα τραπέζι. Παίζεται με μια τράπουλα από 52 φύλλα που είναι χωρισμένα σε 4 σύμβολα (Suits) και καθένα από αυτά έχει 13 φύλλα με διαφορετικές ονομασίες (Ranks).

Ως όρος το πόκερ δεν εννοεί ένα συγκεκριμένο παιχνίδι αλλά υπάρχουν πολλές παραλλαγές αυτού. Η εφαρμογή που αναπτύχτηκε στο πλαίσιο της παρούσας εργασίας βασίζεται στους κανόνες του πιο δημοφιλούς απ' όλες τις παραλλαγές του παιχνιδιού, του Texas Hold 'em.

1.1.1 Εκκίνηση της εφαρμογής

Με το άνοιγμα της εφαρμογής ο χρήστης έρχεται σε επαφή με ένα απλό και λιτό γραφικό περιβάλλον έτσι ώστε η εφαρμογή να είναι εύχρηστη, ευχάριστη και ξεκούραστη. Αρχικά ο χρήστης θα πρέπει να συμπληρώσει τα στοιχεία του στα πλαίσια κειμένου της εφαρμογής για να αποκτήσει πρόσβαση στο παιχνίδι και τις επιλογές του. Μετά την είσοδο του ο χρήστης μεταβαίνει στην οθόνη με τα διάφορα ενεργά τραπέζια ώστε να διαλέξει σε ποιο από αυτά επιθυμεί να συμμετάσχει. Μόλις διαλέξει ποιο τραπέζι τον ικανοποιεί μπαίνει στην οθόνη του τραπεζιού της εφαρμογής, όπου μπορεί να διαλέξει μια ελεύθερη θέση από αυτές που διαθέτει το τραπέζι, αν αυτό είναι δυνατό, και περιμένει την εκκίνηση νέας παρτίδας με την συμμετοχή και του ίδιου σε αυτή.

1.1.2 Σκοπός του παιχνιδιού

Στο συγκεκριμένο τύπο παιχνιδιού, όπως σε όλες τις παραλλαγές του πόκερ, οι παίκτες ανταγωνίζονται για ένα χρηματικό ποσό (μάρκες) στο οποίο συνεισέφεραν οι

ίδιοι. Καθώς οι κάρτες μοιράζονται τυχαία και εκτός του ελέγχου των παικτών, ο κάθε παίκτης προσπαθεί να ελέγξει το χρηματικό ύψος της στοίβας των χρημάτων, ανάλογα με την αξία των καρτών που κρατά. Το παιχνίδι χωρίζεται σε μια σειρά από φάσεις (παρτίδες). Με την ολοκλήρωση της κάθε φάσης τα χρήματα απονέμονται σε έναν ή περισσότερους παίκτες.

Κάθε παρτίδα τελειώνει είτε με το άνοιγμα των καρτών (Showdown) όπου οι εναπομείναντες παίκτες συγκρίνουν τις κάρτες τους, είτε όταν αποσυρθούν (Fold) όλοι οι παίκτες, εγκαταλείποντας την παρτίδα και απομείνει σε αυτή μόνο ένας παίκτης. Στην πρώτη περίπτωση η στοίβα με τις μάρκες δίνεται στον παίκτη ή στους παίκτες που κατέχουν το καλύτερο χαρτί. Συνήθως υπάρχει ένας νικητής, αλλά μπορούν να είναι και περισσότεροι σε περίπτωση ισοπαλίας. Στη δεύτερη περίπτωση, οι μάρκες πηγαίνουν στον παίκτη που δεν εγκατέλειψε την παρτίδα.

Σκοπός των παικτών δεν είναι να κερδίσουν την κάθε παρτίδα, αλλά μάλλον να κάνουν τις μαθηματικά ορθότερες επιλογές. Με αυτές, οι παίκτες αυξάνουν μακροπρόθεσμα τα κέρδη τους, εξασφαλίζοντας το μέγιστο δυνατό κέρδος ή την ελάχιστη δυνατή απώλεια κατά το ποντάρισμα σε κάθε ξεχωριστή παρτίδα.

1.1.3 Επιλογές πονταρίσματος

Αφού έχουν τοποθετηθεί τα τυφλά πονταρίσματα (Blinds) και έχουν μοιραστεί τα φύλλα, αρχίζουν τα πονταρίσματα. Σε αυτή τη φάση και σε κάθε φάση του πονταρίσματος ο χρήστης καλείται να προχωρήσει σε μία από τις παρακάτω επιλογές:

Ντούκου (Check)
- Αν κάποιος άλλος παίκτης δεν έχει ανεβάσει το ποντάρισμα, μπορείς να κάνεις ντούκου, που σημαίνει ότι μένεις στην παρτίδα αλλά δεν ανεβάζεις το ποντάρισμα.

Τα βλέπω (Call)
- Αν κάποιος άλλος παίκτης έχει ανεβάσει το ποντάρισμα, για να μείνεις στην παρτίδα πρέπει να ταιριάξεις, δηλαδή να δεις το ποντάρισμα του.

Πάσο (Fold)
- Αν κάποιος άλλος παίκτης έχει ανεβάσει το ποντάρισμα και θεωρείς ότι δεν αξίζει να μείνεις στην παρτίδα, δεν βλέπεις το ποντάρισμα του και πας πάσο (μένεις εκτός παρτίδας)

Ποντάρισμα (Bet/Raise)
- Τέλος, μπορείς να ανεβάσεις εσύ το ποντάρισμα αν δεν το έχει ήδη κάνει κάποιος άλλος ή και να απαντήσεις στο ποντάρισμα κάποιου άλλου παίκτη προκαλώντας τον με μεγαλύτερο ποντάρισμα.

Πριν πάρει την απόφαση του ο χρήστης, πρέπει να μη ξεχνάει ότι τα φύλλα του τα γνωρίζει μόνο αυτός. Το ίδιο ισχύει για όλους τους παίκτες. Αποτέλεσμα αυτού είναι ότι εκτός από το να παίζει όποτε έχει καλό φύλλο, κάποιος μπορεί να επιλέγει σε ορισμένες παρτίδες να ποντάρει για να δείξει ότι έχει δυνατό φύλλο, ενώ δεν έχει, με

σκοπό να παραπλανήσει τους υπόλοιπους και να τους βγάλει από την παρτίδα. Αυτό λέγεται μπλόφα.

Υπάρχουν, λοιπόν, πολλές παρτίδες που τις κερδίζουν παίκτες που δεν έχουν το καλύτερο φύλλο! Γι αυτόν ακριβώς τον λόγο, εκτός από καλό φύλλο, είναι πολύ σημαντικό στο παιχνίδι να έχει κανείς έξυπνη στρατηγική και να μπορεί να διαβάζει το παιχνίδι των αντιπάλων του. Φυσικά, στις παρτίδες που φτάνουν μέχρι το τέλος και δείχνουν τα φύλλα τους δύο ή παραπάνω παίκτες, κερδίζει αυτός που έχει το καλύτερο φύλλο. Μέχρι εκείνη τη στιγμή όμως (Showdown), η ψυχολογία και η στρατηγική συχνά παίζουν μεγαλύτερο ρόλο από τα φύλλα. Γι αυτό και θεωρείται παιχνίδι πρώτα από όλα επιδεξιότητας, στο οποίο μακροπρόθεσμα οι καλοί παίκτες είναι πάντα κερδισμένοι και οι κακοί χαμένοι.

1.1.4 Κανόνες του πόκερ

Οι κανόνες του παιχνιδιού είναι σχετικά απλοί. Ο καλύτερος συνδυασμός πέντε φύλλων κερδίζει. Σε κάθε παίκτη μοιράζονται δύο κλειστά φύλλα και ο πρώτος γύρος πονταρίσματος ξεκινάει, με τους δύο πρώτους παίκτες στα αριστερά του Ντίλερ να κάνουν δύο υποχρεωτικά και προκαθορισμένα πονταρίσματα. Τα δύο υποχρεωτικά πονταρίσματα ονομάζονται, μικρό τυφλό ποντάρισμα (Small Blind) για τον πρώτο από τα αριστερά παίκτη και μεγάλο τυφλό ποντάρισμα (Big Blind) για τον δεύτερο από αριστερά.

Στη συνέχεια ο Ντίλερ «καίει» ένα φύλλο και βγάζει τρία ανοιχτά φύλλα στο κέντρο του τραπεζιού (Flop) τα οποία είναι διαθέσιμα σε όλους τους παίκτες. Ο δεύτερος γύρος πονταρίσματος ξεκινάει, αυτή τη φορά χωρίς τα τυφλά πονταρίσματα. Όταν όλοι οι παίκτες ολοκληρώσουν τις κινήσεις τους ο Ντίλερ «καίει» ακόμα ένα φύλλο και βγάζει τέταρτο ανοιχτό φύλλο στο τραπέζι (Turn) και ξεκινάει ο τρίτος γύρος πονταρίσματος. Το παιχνίδι τελειώνει όταν ο Ντίλερ μοιράσει και το πέμπτο ανοιχτό φύλλο (River) με την ίδια διαδικασία και ολοκληρωθεί και ο τέταρτος γύρος πονταρίσματος. Νικητής είναι ο παίκτης που έχει τον καλύτερο συνδυασμό 5 φύλλων από τα 7 που έχει διαθέσιμα.

1.1.5 Συνδυασμοί φύλλων

Παρακάτω δίνονται οι συνδυασμοί φύλλων σε φθίνουσα σειρά, δηλαδή οι καλύτεροι συνδυασμοί αναφέρονται πρώτα.

Φλος Ρουαγιάλ (Royal Flush): Φλος που καταλήγει στον Άσσο (π.χ. δέκα, βαλές, ντάμα, ρήγας, άσσος του ίδιου χρώματος).

Φλος (Straight Flush): Κέντα με κάρτες του ίδιου χρώματος.

Καρέ (Four of a Kind): Τέσσερις όμοιες κάρτες, διαφορετικού χρώματος (π.χ. τέσσερα εννιάρια).

Φουλ (Full House): Τρία όμοια χαρτιά (τρία εννιάρια διαφορετικού χρώματος) με άλλα δυο όμοια (δυο οχτάρια)

Χρώμα (Flush): Πέντε χαρτιά ανόμοια αλλά του ίδιου χρώματος (π.χ. κούπες, μπαστούνια, τριφύλλια, καρό)

Κέντα (Straight): Πέντε χαρτιά σε σειρά. Για παράδειγμα 4,5,6,7,8 βέβαια όχι του ίδιου χρώματος.
Τρία Όμοια (Three of a Kind): Τρεις όμοιες κάρτες, διαφορετικού χρώματος.
Δύο Ζεύγη (Pairs, Two Pair): Δύο ζευγάρια όμοιων φύλλων, για παράδειγμα βαλές βαλές και δέκα δέκα.
Ένα Ζεύγος (Pair): Ένα ζευγάρι όμοιων φύλλων.
Μεγαλύτερο Φύλλο (High Card): Στην περίπτωση που δεν προκύπτει κάποιος από τους προηγούμενους συνδυασμούς, τότε νικητής της παρτίδας είναι αυτός που έχει το μεγαλύτερο φύλλο. Για παράδειγμα τον άσσο.

1.2 Γλώσσα Προγραμματισμού C# και .NET

1.2.1 Τι είναι το .NET

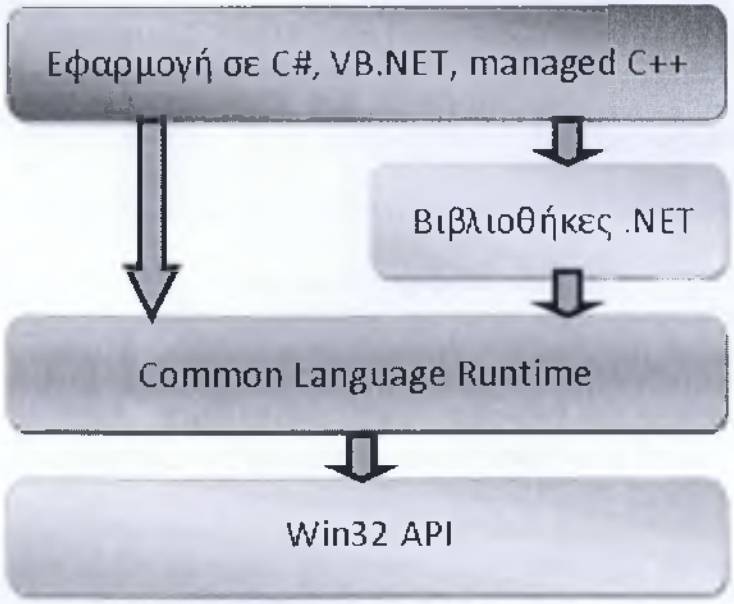
Το .NET είναι μια νέα πλατφόρμα ανάπτυξης εφαρμογών σε περιβάλλοντα Windows. Σύμφωνα με τη Microsoft, έχει ως σκοπό την απλοποίηση της ανάπτυξης εφαρμογών «κρύβοντας» τις τεχνικές λεπτομέρειες υλοποίησης πολλών λειτουργιών, όπως διαχείριση μνήμης, επικοινωνία μέσω δικτύου, είσοδο/έξοδο από συσκευές και αφήνοντας τον προγραμματιστή ελεύθερο να επικεντρωθεί στη «λογική» του προγράμματος.

Το .NET χαρακτηρίζεται ως managed πλατφόρμα με την έννοια ότι δημιουργεί ένα ελεγχόμενο και ασφαλές περιβάλλον μέσα στο οποίο μπορεί να τρέξει μια εφαρμογή. Η ασφάλεια έγκειται για παράδειγμα στον έλεγχο στην δέσμευση και προσπέλαση της μνήμης (δεν υπάρχουν pointers, δεν μπορείς να προσπελάσεις μια θέση μνήμης εκτός πίνακα), στον τύπο των μεταβλητών και δεδομένων (δεν μπορείς να θέσεις μια float τιμή σε μια ακέραια μεταβλητή) ή στην αυτόματη υλοποίηση δικλείδων ασφαλείας.

Το .NET υποστηρίζει πληθώρα γλωσσών προγραμματισμού οι οποίες είναι ειδικά σχεδιασμένες για αυτό, όπως C#, Visual Basic.NET, J++ και managed C++. Στην πραγματικότητα, το .NET καταλαβαίνει μόνο μια γλώσσα προγραμματισμού την Microsoft Intermediate Language (MSIL ή CIL[1]). Συνεπώς, οποιαδήποτε γλώσσα προγραμματισμού μπορεί να μεταγλωττιστεί σε MSIL μπορεί να τρέξει στην πλατφόρμα .NET. Ο χρήστης μπορεί ακόμα να γράψει απευθείας ένα πρόγραμμα σε MSIL στο Notepad να το κάνει μεταγλώττιση και να το τρέξει στο .NET!

Καρδιά του .NET αποτελεί το λεγόμενο Common Language Runtime (CLR). Η οντότητα αυτή είναι το managed περιβάλλον μέσα στο οποίο τρέχουν οι εφαρμογές .NET. Κατά μια έννοια κρύβει το δύσχρηστο Win32 API που χρησιμοποιείται συχνά για προγραμματισμό εφαρμογών Windows και παρουσιάζει στο χρήστη ένα απλούστερο και περισσότερο εύχρηστο (βλέπε παρακάτω σχήμα).

[1] Microsoft Intermediate Language ή Common Intermediate Language: είναι το χαμηλότερο επίπεδο της ανθρώπινα αναγνώσιμης γλώσσας όπως αυτή ορίζεται από τον Common Language Infrastructure προσδιορισμό, και χρησιμοποιείται από το .Net Framework και το Mono Project.



Σχήμα 1.2-1: Γραφική αναπαράσταση του .Net

Παράλληλα το .NET παρέχει και μια πληθώρα κλάσεων για κάθε δουλειά που είναι πολύ καλά οργανωμένες σε χώρους ονομάτων (namespaces), έτσι ώστε με λίγη εξοικείωση να βρίσκεις την κλάση που σου χρειάζεται.

Ο κώδικας που εκτελείται στο CLR έρχεται υπό μορφή assemblies, με επέκταση .dll ή .exe (δεν έχουν σχέση με τα κλασσικά .dll αρχεία των Windows). Το .NET, όπως και η Java χαρακτηρίζονται από την λεγόμενη Just In Time μεταγλώττιση. Ο κώδικας, στη γλώσσα προγραμματισμού που χρησιμοποιεί ο χρήστης, μεταγλωττίζεται αρχικά σε MSIL η οποία αποθηκεύεται σε ένα εκτελέσιμο .exe αρχείο ή σε μια βιβλιοθήκη .dll. Όταν ο χρήστης τρέξει το πρόγραμμα που ανέπτυξε, το CLR διαβάζει το MSIL κώδικα του αρχείο και Just In Time (JIT) το μεταγλωττίζει σε κώδικα Windows (native) έτοιμο προς εκτέλεση, και στην συνέχεια τον εκτελεί. Αυτό το επιπλέον βήμα πριν την εκτέλεση του κώδικα διαφοροποιεί μια managed εφαρμογή σε .NET από μια unmanaged σε C++ για παράδειγμα. Το πρόγραμμα σε C++ είναι ήδη μεταγλωττισμένο σε native Windows κώδικα και τρέχει απευθείας.

Ένα ακόμα χαρακτηριστικό του CLR είναι η αυτοματοποιημένη διαχείριση μνήμης. Σε κλασσικές γλώσσες προγραμματισμού (unmanaged) όπως η C++, όταν ο χρήστης δεσμεύσει μια ποσότητα μνήμης για να αποθηκεύσει ένα αντικείμενο πρέπει να είναι πολύ προσεκτικός στο να την αποδεσμεύσει, να την επιστρέψει στο σύστημα δηλαδή, όταν δεν τη χρειάζεται. Αν το αγνοήσει αυτό συστηματικά, τότε θα δημιουργηθεί το επονομαζόμενο memory leak, δηλαδή η διαθέσιμη μνήμη του συστήματος θα ελαττώνεται διαρκώς και σε κάποιο σημείο τα Windows θα στερέψουν από ελεύθερη μνήμη.

Αντιθέτως, το CLR προσφέρει ένα μηχανισμό Garbage Collection. Ο χρήστης μπορεί να ζητήσει όση μνήμη χρειάζεται από το σύστημα και να μην ασχοληθεί με την

απελευθέρωση της. Ο Garbage Collector υλοποιεί μηχανισμούς που του επιτρέπουν να «καταλάβει» πότε μια δεσμευμένη ποσότητα μνήμης δεν χρησιμοποιείται πλέον και αυτόματα την απελευθερώνει για μετέπειτα χρήση.

1.2.2 Πλεονεκτήματα .NET

Το .NET έχει πολλά πλεονεκτήματα για την ανάπτυξη εφαρμογών:

- Είναι εγγενώς αντικειμενοστραφής πλατφόρμα.
- Είναι ανεξάρτητο από γλώσσα προγραμματισμού. Σε μια εφαρμογή ένας προγραμματιστής μπορεί να γράφει κώδικα σε C#, άλλος σε VB.NET και άλλος σε managed C++ και τα τμήματα που αναπτύσσει ο καθένας να συνεργάζονται μεταξύ τους χωρίς προβλήματα.
- Η χρήση βιβλιοθηκών (assemblies) κάνει πολύ εύκολη την επαναχρησιμοποίηση κώδικα.
- Παρέχει πολύ εύκολη εγκατάσταση. Αρκεί να αντιγράψουμε το κατάλογο της εφαρμογής σε ένα άλλο υπολογιστή και αυτή θα τρέξει άμεσα. Δεν υπάρχει διαδικασία εγκατάστασης, δεν πειράζει το registry.
- Παρέχει πληθώρα έτοιμων λειτουργιών που κάνουν την ανάπτυξη κώδικα πολύ εύκολη.
- Αυτοματοποιημένη διαχείριση μνήμης, ο χρήστης δεν χρειάζεται να ασχοληθεί με αποδέσμευση μνήμης.

1.2.3 Μειονεκτήματα .NET (για ανάπτυξη παιχνιδιών)

Το .NET έχει δυο μειονεκτήματα που αφορούν ειδικά την ανάπτυξη βιντεοπαιχνιδιών και όχι την γενική ανάπτυξη εφαρμογών:

- Αυτοματοποιημένη διαχείριση μνήμης, ο χρήστης δεν χρειάζεται να ασχοληθεί με αποδέσμευση μνήμης.
- Το CLR εισάγει μια (μικρή ίσως) καθυστέρηση στην εκτέλεση της εφαρμογής.

Το πρώτο, αν και είναι πολύ χρήσιμο χαρακτηριστικό για γενικό προγραμματισμό, σε ένα βιντεοπαιχνίδι μπορεί να δημιουργήσει πρόβλημα. Η απελευθέρωση μνήμης που δεν χρησιμοποιείται πλέον είναι μια ακριβή εργασία (από πλευράς χρόνου), η οποία επιπλέον είναι μη-ντετερμινιστική, μπορεί να συμβεί δηλαδή οποτεδήποτε. Αυτό μπορεί να έχει σαν αποτέλεσμα ένα παιχνίδι που τρέχει σε ένα σταθερό ρυθμό 60 καρέ το δευτερόλεπτο, να δει μια δραματική πτώση στο ρυθμό ανανέωσης για 1 δευτερόλεπτο, πράγμα ανεπίτρεπτο στην ανάπτυξη βιντεοπαιχνιδιών.

Το δεύτερο αφορά στη Just In Time μεταγλώττιση που υποστηρίζει το CLR. Από τη μία εισάγει μια καθυστέρηση στην εκκίνηση του παιχνιδιού μιας και πρέπει να μεταγλωττιστεί ο κώδικας από την άλλη ο μεταγλωττιστής ο ίδιος δεν είναι βέλτιστος με την έννοια ότι δεν παράγει το καλύτερο δυνατό native κώδικα για Windows. Και τα δυο προβλήματα αυτά μπορούν να αντιμετωπιστούν αν τα λάβουμε υπόψη κατά την ανάπτυξη της εφαρμογής.

1.2.4 Τι είναι η C#

Η C# είναι μια αντικειμενοστραφής γλώσσα υψηλού επιπέδου η οποία φτιάχτηκε από τη Microsoft. Δανείζεται πολλά στοιχεία, και έχει παρόμοια σύνταξη, με την C++ και την Java, κάνοντας έτσι την εκμάθηση της σχετικά εύκολη για άτομα με εμπειρία σε αυτές τις γλώσσες. Φτιάχτηκε με σκοπό να αποτελέσει την κύρια γλώσσα ανάπτυξης στην πλατφόρμα .NET. Είναι γλώσσα ειδικά σχεδιασμένη για να υποστηρίζει το .NET Framework της ίδιας εταιρείας. Βασικό χαρακτηριστικό της είναι ότι δεν παράγει απευθείας κώδικα μηχανής όπως η C++, άλλα έναν ενδιάμεσο κώδικα που στοχεύει το .NET. Η συγκεκριμένη γλώσσα είναι τόσο μεταγλωττιζόμενη (compiled language) όσο και διερμηνευόμενη (interpreted language) και είναι μια από τις πολλές γλώσσες της Microsoft που υλοποιούν το πρότυπο CLI[2].

1.3 Βάση Δεδομένων

Στο πρόγραμμα μας χρειάστηκε να χρησιμοποιήσουμε μια βάση δεδομένων για να αποθηκεύονται τα στοιχεία των χρηστών. Τα δεδομένα τα οποία χρειάζεται να αποθηκεύουμε στην παρούσα φάση είναι λίγα (μόλις ένας πίνακας με μικρό αριθμό γραμμών). Θα μπορούσαμε λοιπόν να κάνουμε χρήση της βάσης δεδομένων MDB (Microsoft Data Base- του Microsoft Access), όμως αποφασίσαμε να χρησιμοποιήσουμε τον SQL Server της Microsoft, καθ' ότι ο SQL Server είναι η πλέον ενδεδειγμένη λύση για την χρήση βάσης δεδομένων σε εφαρμογές γραμμένες στη γλωσσά προγραμματισμού C#.

Επειδή τόσο η βάση δεδομένων όσο και η γλώσσα προγραμματισμού που χρησιμοποιηθήκαν για την ανάπτυξη της εφαρμογής μας είναι τεχνολογίες της εταιρείας Microsoft δε χρειάζεται η διενέργεια κάποιας ιδιαίτερης διαδικασίας από τον χρήστη, για την συνεργασία των δυο αυτών εργαλείων. Από την στιγμή που τα προγράμματα είναι σωστά εγκατεστημένα, το μόνο που χρειάζεται για την χρήση της βάσης δεδομένων είναι η γνώση της γλώσσας SQL.

1.4 Εργαλείο XNA

1.4.1 Ιστορική αναδρομή

Το XNA είναι ένα σύνολο εργαλείων της Microsoft «διαχειριζόμενου» κώδικα, από ένα περιβάλλον εκτέλεσης, το όποιο διευκολύνει την ανάπτυξη και διαχείριση των βιντεοπαιχνιδιών. Το XNA επιχειρεί να απελευθερώσει τους προγραμματιστές παιχνιδιών από το να γράφουν «βασικό» κώδικα και συγκεντρώνει διαφορετικές προοπτικές για την παραγωγή παιχνιδιών σε ένα σύστημα. Το XNA εργαλείο ανακοινώθηκε στις 24 Μάρτιου του 2004, στο συνέδριο ανάπτυξης παιχνιδιών, στο Σαν Χοσέ (Καλιφόρνια). Η πρώτη επαφή με το ευρύ κοινό έγινε στις 14 Μαρτίου του 2006 με την πρώτη έκδοση του XNA Game Studio, τον Δεκέμβριο του 2007 κυκλοφόρησε το

[2] Common Language Infrastructure: πρότυπο ορισμένο από τη Microsoft και τυποποιημένο από τα ISO και ECMA. Περιγράφει τον εκτελέσιμο κώδικα και το περιβάλλον εκτέλεσης που σχηματίζει τον πυρήνα του .Net Framework και των υλοποιήσεων ανοιχτού κώδικα Mono Project και Portable.Net.

XNA Game Studio 2, ακολουθούμενο από το XNA Game Studio 3 το οποίο και κυκλοφόρησε στις 30 Οκτώβριου του 2008. Στις 16 Σεπτεμβρίου του 2010 κυκλοφόρησε η έκδοση του XNA Game Studio 4 περιλαμβάνοντας και το Windows Phone Development Tools, το οποίο περιέχει εργαλεία για την ανάπτυξη εφαρμογών για το λειτουργικό των Windows Phone. Το XNA αυτή την στιγμή περιλαμβάνει όλο το τμήμα ανάπτυξης παιχνιδιών της Microsoft, περιλαμβάνοντας το Xbox Development Kit (το οποίο είναι το βασικό εργαλείο για την ανάπτυξη εφαρμογών για την κονσόλα του Xbox) και το XNA Game Studio για την ανάπτυξη εφαρμογών για λειτουργικά συστήματα της Microsoft. Η λέξη "XNA" προέρχεται από το όνομα ανάπτυξης του έργου, *Xbox New Architecture*, με την μετονομασία του Xbox σε Xbox 360(2005), το όνομα XNA πλέον προέρχεται από το XNA's Not Acronymed.

1.4.2 XNA Framework

Το XNA Framework είναι βασισμένο στην υλοποίηση του .NET Compact Framework 2.0 για την ανάπτυξη εφαρμογών για το Xbox 360 και του .NET Framework 2.0 για την υλοποίηση εφαρμογών για τα Windows. Περιλαμβάνει ένα επεκταμένο σύνολο από βιβλιοθήκες κλάσεων, ειδικές για την δημιουργία παιχνιδιών, που προωθούν την μεγίστη επαναχρησιμοποίηση κώδικα. Τα παιχνίδια που είναι υλοποιημένα με το XNA μπορούν να «τρέξουν» σε οποιαδήποτε πλατφόρμα υποστηρίζει το XNA Framework με ελάχιστες ή και καθόλου διαφοροποιήσεις. Επίσης τα παιχνίδια τα οποία «τρέχουν» μέσω του Framework μπορούν τεχνικά να γραφτούν σε οποιαδήποτε .NET συμβατή γλώσσα.

Επιπλέον το XNA Framework ενθυλακώνει τεχνικές λεπτομέρειες χαμηλού επιπέδου και εργαλεία που περιέχονται στην δημιουργία ενός παιχνιδιού, διασφαλίζοντας ότι το Framework θα φροντίσει τις διαφορές που προκύπτουν όταν ένα παιχνίδι μεταφέρεται από μια συμβατή πλατφόρμα σε μια άλλη, επιτρέποντας στους προγραμματιστές να εστιάζουν περισσότερο στο περιεχόμενο του παιχνιδιού. Το XNA Framework συνεργάζεται μέ μια σειρά εργαλείων, για να βοηθήσει στην δημιουργία των περιεχόμενων του παιχνιδιού. Το XNA Framework υποστηρίζει τόσο δισδιάστατα (2D) όσο και τρισδιάστατα (3D) γραφικά για την δημιουργία παιχνιδιών, ενώ επιτρέπει τη χρήση χειριστηρίων της κονσόλας του Xbox 360 και της λειτουργίας δόνησης αυτών. Τα XNA παιχνίδια που υλοποιούνται για τις κονσόλες του Xbox μπορούν να διανέμονται προς το παρόν μόνο στα μέλη του Microsoft XNA Creator's Club (ετήσια χρέωση για τα μέλη της), ενώ οι εφαρμογές που αναπτύσσονται για τα υπόλοιπα λειτουργικά συστήματα, μπορούν να διανέμονται δωρεάν κάτω από τους όρους χρήσης της Microsoft.

1.4.3 XNA Build

Το XNA Build είναι ένα σύνολο από εργαλεία διαχείρισης των προϊόντων διασωλήνωσης (pipeline) των παιχνιδιών, που βοηθούν με τον καθορισμό, τη συντήρηση, την αποσφαλμάτωση και τη βελτιστοποίηση των προσπαθειών ανάπτυξης αυτόνομων παιχνιδιών. Ένα προϊόν διασωλήνωσης παιχνιδιού περιγράφει την

διεργασία με την οποία το περιεχόμενο του παιχνιδιού, όπως γραφικά και τρισδιάστατα μοντέλα, μετασχηματίζονται σε μια μορφή κατάλληλη για χρήση από τις μηχανές παιχνιδιών. Το XNA Build αναγνωρίζει τις εξαρτήσεις της διασωλήνωσης (pipeline) και παρέχει πρόσβαση σε ένα **A**pplication **P**rogram **I**nterface[3] που ενεργοποιεί την περαιτέρω επεξεργασία των εξαρτώμενων δεδομένων. Τα εξαρτώμενα δεδομένα μπορούν να αναλυθούν έτσι ώστε να βοηθήσουν στην ελάττωση του μεγέθους του παιχνιδιού εντοπίζοντας περιεχόμενα που στην πραγματικότητα δεν χρησιμοποιούνται. Για παράδειγμα, μια XNA Build ανάλυση έδειξε ότι το 40% των γραφικών που αποστέλλονται στο MechCommander 2 δεν χρησιμοποιούνται και θα μπορούσαν να είχαν παραληφθεί.

1.4.4 XNA Game Studio

Το XNA Game Studio Express είναι η πρώτη έκδοση του XNA Game Studio[4]. Προορίζεται για σπουδαστές, ερασιτέχνες και ανεξάρτητους προγραμματιστές παιχνιδιών και η διανομή του γίνεται δωρεάν. Αυτή η έκδοση του προγράμματος παρέχει τα βασικά «αρχικά εργαλεία» για γρήγορη ανάπτυξη συγκεκριμένων παιχνιδιών, όπως τα παιχνίδια πλατφόρμας (Platform Games), τα παιχνίδια στρατηγικής πραγματικού χρόνου (Real-Time Strategy) και τα παιχνίδια βολών πρώτου προσώπου (First Person Shooter). Οι προγραμματιστές μπορούν να δημιουργούν παιχνίδια με το XNA Framework για την πλατφόρμα των Windows χωρίς κόστος, όμως για να «τρέξουν» τα παιχνίδια τους στην πλατφόρμα του Xbox 360 θα πρέπει να πληρώσουν μια ετήσια αμοιβή ή συνδρομή τεσσάρων μηνών για να αποκτήσουν πρόσβαση στο Microsoft XNA Creator's Club. Στην κυκλοφορία της πρώτης έκδοσης δεν υπήρχε τρόπος αποστολής προ-μεταγλωττισμένων δυαδικών αρχείων σε άλλους χρήστες του Xbox 360, βέβαια αυτό διορθώθηκε με την κυκλοφορία του XNA Game Studio Express 1.0 Refresh που έδωσε τη δυνατότητα μεταγλώττισης δυαδικών αρχείων του Xbox 360 και διαμοιρασμού αυτών στους χρήστες του Microsoft XNA Creator's Club. Η πρώτη δοκιμαστική έκδοση κυκλοφόρησε στις 30 Αυγούστου του 2006, ακολουθουμένη από την δεύτερη έκδοση στις 1 Νοεμβρίου του 2006. Η τελική έκδοση κυκλοφόρησε στις 11 Δεκεμβρίου του 2006. Στις 24 Απριλίου του 2007 κυκλοφόρησε η αναβάθμιση υπό την ονομασία XNA Game Studio Express 1.0 Refresh.

Έπειτα κυκλοφόρησε το XNA Game Studio 2.0 στις 13 Δεκεμβρίου του 2007. Τα χαρακτηριστικά του XNA Game Studio 2.0 είναι η ικανότητα του να χρησιμοποιείται με όλες τις εκδόσεις του Virtual Studio 2005, ένα δικτυακό πρόγραμμα διεπαφής [εφαρμογής (API)] που χρησιμοποιεί το Xbox Live τόσο στην κονσόλα του Xbox 360 όσο και στις πλατφόρμες των Windows και τέλος καλύτερη διαχείριση συσκευών.

Ακολούθησε το XNA Game Studio 3.0 (για όλες τις εκδόσεις του Virtual Studio 2008) που επιτρέπει τη δημιουργία παιχνιδιών που στοχεύουν στις πλατφόρμες Zune και στη δημιουργία της κοινότητας υποστήριξης του Xbox Live. Η δοκιμαστική έκδοση

[3] **A**pplication **P**rogram **I**nterface: Ένα σύνολο κοινών λειτουργιών και υπορουτίνων που χρησιμοποιούνται για την εκτέλεση ενός συγκεκριμένου σκοπού σε ένα πρόγραμμα.
[4] Το XNA Game Studio είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) για την δημιουργία παιχνιδιών.

των εργαλείων κυκλοφόρησε τον Σεπτέμβριο του 2008. Η τελική του έκδοση κυκλοφόρησε στις 30 Οκτώβριου του 2008. Το XNA Game Studio 3.0 τώρα υποστηρίζει την C# 3.0, LINQ. Επίσης υπάρχουν μερικές ακόμα καινούργιες λειτουργίες, όπως μια μέθοδος δοκιμαστικής έκδοσης (Trial Mode) που προστίθεται στο XNA Game Studio 3.0 και επιτρέπει στους δημιουργούς των εφαρμογών να μπορούν με μεγάλη ευκολία να προσθέσουν τη λειτουργία της δοκιμαστικής έκδοσης στα προγράμματα τους. Ακόμα εμφανίζονται λειτουργίες που αφορούν στα παιχνίδια πολλών παικτών στο Xbox Live όπως τις προσκλήσεις μέσα από το παιχνίδι και την δημιουργία παιχνιδιών ανεξάρτητης πλατφόρμας (Cross-Platform) τα οποία «τρέχουν» σε Windows, Xbox 360 και Zune.

Μια από της τελευτές εκδώσεις είναι το XNA Game Studio 3.1 το οποίο κυκλοφόρησε στις 11 Ιουνίου του 2009. Το πρόγραμμα διεπαφής [εφαρμογής (API)] συμπεριλαμβάνει την υποστήριξη της αναπαραγωγής βίντεο καθώς και ένα αναθεωρημένο πρόγραμμα διεπαφής εφαρμογής ήχου, το σύστημα Xbox Live Party και υποστήριξη προκειμένου να μπορούν τα παιχνίδια να χρησιμοποιούν την λειτουργία του Xbox 360 Avatars.

Τέλος, το XNA Game Studio 4.0 ανακοινώθηκε και κυκλοφόρησε αρχικά ως Community Technical Preview[5] στο συνέδριο προγραμματιστών παιχνιδιών (GDC) στις 9 Μαρτίου του 2010, και στην τελική του μορφή στις 16 Σεπτεμβρίου 2010. Προστίθεται υποστήριξη για την πλατφόρμα των Windows Phone (περιλαμβάνοντας τρισδιάστατη επιτάχυνση υλικού), πλαίσιο εργασίας του προφίλ υλικού (hardware profiles), προσαρμοζόμενα εφέ, ενσωματωμένα αντικείμενα κατάστασης (state objects), τους συντελεστές της μηχανής γραφικών, προσανατολισμός, ανεξαρτησία πλατφόρμας, είσοδος από οθόνη πολλαπλής αφής, είσοδος από μικρόφωνο, αναπαραγωγή ήχου ενδιάμεσης αποθήκευσης και ενσωμάτωση στο Visual Studio 2010.

1.4.5 XDK Extensions

Γνωστό ως XNA Game Studio Professional, το XDK Extensions είναι ένα πρόσθετο του XNA Game Studio και χρειάζεται το Microsoft Xbox 360 Development Kit. Και τα δυο είναι διαθέσιμα για προγραμματιστές του Xbox που διαθέτουν την κατάλληλη άδεια. Το XDK Extensions περιλαμβάνει επιπλέον διαχειριζόμενα προγράμματα διεπαφής [εφαρμογής (API)] για τα κατορθώματα, πίνακες και άλλες λειτουργίες δεσμευμένες για τίτλους παιχνιδιών με άδεια χρήσης. Οι τίτλοι παιχνιδιών που αναπτύσσονται με την χρήση του XDK Extensions μεταξύ άλλων συμμετέχουν στον διαγωνισμό Dream-Build-Play της Microsoft. Το πιο δημοφιλές από αυτά είναι το "The Dishwasher: Dead Samurai".

[5] Community Technical Preview (CTP): Μια από τις φάσεις δοκιμών μιας εφαρμογής- προγράμματος ορισμένη από την Microsoft.

1.4.6 Άδεια χρήσης

Το Microsoft XNA Framework 2.0 EULA[6] απαγορεύει τη διανομή δικτυακών εμπορικών παιχνιδιών που συνδέονται με το Xbox Live και το Games for Windows Live λόγω έλλειψης ειδικής συμφωνίας μεταξύ του προγραμματιστή και της Microsoft. Αυτό σημαίνει ότι το XNA Game Studio μπορεί να χρησιμοποιηθεί για την ανάπτυξη εμπορικών παιχνιδιών και άλλων προγραμμάτων για την πλατφόρμα των Windows, παρ' όλο που ο κώδικας δικτυακής υποστήριξης της Microsoft για Xbox/Windows Live δεν μπορεί να χρησιμοποιηθεί. Δικτυακός κώδικας τρίτων μπορεί να χρησιμοποιηθεί στην ανάπτυξη ενός έργου XNA. Τα παιχνίδια που αναπτύσσονται με την χρήση του XNA Game Studio μπορούν να διανεμηθούν μέσω του Xbox Live Indie Games και του Windows Phone Marketplace.

1.4.7 Xbox Live Indie Games

Τα παιχνίδια που είναι γραμμένα για το Xbox 360 μπορούν να υποβάλλονται στο AppHub, για το όποιο απαιτείται ετήσια συνδρομή. Όλα τα παιχνίδια που υποβάλλονται στο AppHub υπόκεινται σε αξιολόγηση από άλλους δημιουργούς. Αν το παιχνίδι περάσει την αξιολόγηση τότε καταχωρείται στο Xbox Live Marketplace. Οι δημιουργοί μπορούν να θέσουν μια τιμή των 80, 240 ή 400 πόντων για το παιχνίδι τους. Ο δημιουργός του πληρώνεται με το 70% των συνολικών εσόδων από τις πωλήσεις του παιχνιδιού. Η Microsoft αρχικά είχε ως σκοπό να εισπράττει ένα επιπλέον ποσό από τα έσοδα σε περίπτωση που παρείχε επιπλέον προώθηση για το παιχνίδι, όμως αυτή η πολιτική ανακλήθηκε τον Μάρτιο του 2009, εισπράττοντας σταθερά το 30% των εσόδων ανεξάρτητα από την διαφήμιση του παιχνιδιού. Επίσης η Microsoft διανέμει δωρεάν συνδρομή στο AppHub ενός χρόνου για εκπαιδευτικά ιδρύματα μέσω του DreamSpark και του MSDNAA. Μέσω αυτών των λογαριασμών μπορούν οι σπουδαστές να αναπτύξουν τα δικά τους παιχνίδια για το Xbox 360. Βέβαια οι σπουδαστές χρειάζονται ακόμα έναν λογαριασμό στο Xbox Live προκειμένου να μπορούν να προωθήσουν το παιχνίδι τους στην αγορά.

1.4.8 Εναλλακτικές υλοποιήσεις

Ένα έργο με όνομα Mono.XNA σχηματίστηκε με σκοπό την δημιουργία ενός Ecma International προτύπου συνόλου εργαλείων ανοιχτού κώδικα συμβατά με το .Net Framework που συμπεριλαμβάνουν μεταξύ άλλων, C# μεταγλωττιστή και του Common Language Runtime. Ο σκοπός που ξεκίνησε το έργο δεν ήταν μόνο η δυνατότητα εκτέλεσης εφαρμογών του .Net Framework της Microsoft ανεξαρτήτου πλατφόρμας αλλά και το να παραδώσουν καλύτερα προγραμματιστικά εργαλεία στους προγραμματιστές της πλατφόρμας των Linux. Το Mono.XNA μπορεί να «τρέξει» σε πάρα πολλά λειτουργικά συστήματα όπως το Android, BSD, iOS, Linux, Mac OS X, Windows, Solaris, Unix και σε κονσόλες παιχνιδιών όπως το PlayStation 3, Wii και το

[6] End-User License Agreement (Άδεια χρήσης τελικού χρήστη): Στη βιομηχανία του ιδιόκτητου λογισμικού ή του λογισμικού κλειστού κώδικα άδεια χρήσης τελικού χρήστη είναι μια σύμβαση μεταξύ του παρόχου άδειας χρήσης και του αγοραστή-χρήστη του λογισμικού, καθορίζοντας τα δικαιώματα χρήσης του από αυτόν.

Xbox 360. Από την βάση του κώδικα του Mono.XNA και του SilverSprite ένα νέο έργο αναπτύχτηκε με όνομα MonoGame το οποίο είχε στόχους παρόμοιους με το Mono.XNA. Σήμερα το MonoGame παρέχει σταθερή υποστήριξη για τις πλατφόρμες των iOS και των Mac OS X με περιορισμένη υποστήριξη για τις πλατφόρμες των Linux, Windows, Android, καθώς και την σχεδιαστική γλώσσα Metro και το πλαίσιο λειτουργίας PlayStation Mobile. Επίσης ένα έργο ανοιχτού κώδικα με όνομα Grommet περιλαμβάνει περιορισμένη μεταφορά του .NET Micro Framework για ενσωματωμένες συσκευές. Τέλος υπάρχει και το ANX το οποίο είναι ένα έργο που υλοποιεί την δικιά του εκδοχή του XNA χρησιμοποιώντας το SharpDX[7]. Με το ANX οι προγραμματιστές μπορούν να γράφουν παιχνίδια χρησιμοποιώντας κώδικα που είναι σχεδόν ίδιος με το XNA, ενώ στην πλατφόρμα των Windows 8 θεωρείται εφαρμογή τύπου Metro.

1.5 Neoforce XNA Library

Το Neoforce είναι μια βιβλιοθήκη συμβατή με την τεχνολογία XNA 4.0 η οποία υλοποιεί την δική της εκδοχή των βασικών στοιχείων ελέγχου της διεπαφής χρήστη (User Interface). Το έργο αποτελεί έμπνευση του Tom Shane, ο οποίος μαζί με μια μικρή ομάδα προγραμματιστών ανέλαβαν την ανάπτυξη και συντήρηση του έργου. Τον Σεπτέμβριο του 2010 κυκλοφόρησε η τελευταία δοκιμαστική (Beta) έκδοση της βιβλιοθήκης στο ευρύ κοινό, ταυτόχρονα ανακοινώθηκε και η διακοπή της περαιτέρω ανάπτυξης του έργου από τον Tom Shane και της ομάδας του. Το έργο έγινε «ανοιχτού» κώδικα (open source) και δόθηκε η ελευθερία στους χρήστες για κάθε χρήση και επέμβαση στον κώδικα. Δυστυχώς στον κώδικα δεν υπάρχουν καθόλου σχόλια ενώ και η βιβλιογραφία που απέμεινε από την ομάδα ανάπτυξης ήταν ελάχιστη. Η μοναδική και ουσιαστική βοήθεια ως προς την χρήση της βιβλιοθήκης αποτελούν κάποια παραδείγματα τα οποία υλοποιηθήκαν για δοκιμαστικούς λογούς από τους προγραμματιστές και με την παύση ανάπτυξης δόθηκαν στο κοινό μαζί με τον κώδικα της βιβλιοθήκης, ενώ έγιναν προσπάθειες ανάλυσης του κώδικα και από άλλους χρήστες. Όμως δεν υπάρχει επίσημα κάποια ολοκληρωμένη δουλεία παρά μόνο κάποιες αναρτήσεις σε προσωπικά μπλογκ που παρέχουν μια εισαγωγική γνώση και ανάλυση για τεχνικές που χρησιμοποιήθηκαν στην ανάπτυξη της βιβλιοθήκης.

Ένα ιδιαίτερο χαρακτηριστικό της βιβλιοθήκης είναι η «εξαναγκαστική» χρήση αρχείων εμφάνισης (Skins) τα οποία περιέχουν πληροφορίες για την εμφάνιση των στοιχείων ελέγχου (Controls). Η βιβλιοθήκη υλοποιεί και παρέχει στον χρήστη προεπιλεγμένα αρχεία εμφάνισης με ρυθμίσεις για τα βασικά στοιχεία ελέγχου και παράλληλα δίνει την δυνατότητα στους χρήστες για την δημιουργία προσαρμοσμένων αρχείων εμφάνισης για εξ ολοκλήρου ή τμηματική χρήση σε ορισμένα στοιχεία ελέγχου. Με αυτόν τον τρόπο παρέχεται στους χρήστες της βιβλιοθήκης η δυνατότητα να επιλέγουν το στυλ και την εμφάνιση των στοιχείων ελέγχου ανάλογα με το ύφος της εφαρμογής που πρόκειται να χρησιμοποιηθούν ή ακόμα και να δημιουργούν μια γκάμα από αρχεία εμφάνισης έτσι ώστε ο κάθε χρήστης της εφαρμογής να επιλέγει και να

[7] SharpDX: είναι ένα ανοιχτού κώδικα διαχειριζόμενο DirectX API αυτόνομης πλατφόρμας για το περιβάλλον του .Net. Το συγκεκριμένο API αυτό-δημιουργείται απευθείας από τα DirectX C++ SDK Headers και ο παραγόμενος κώδικας είναι καθαρά στη γλώσσα C# χωρίς καμία χρήση C++/CLI assemblies, πετυχαίνοντας παρ' όλα αυτά συμβατή απόδοση.

«δίνει» στο πρόγραμμα την εμφάνιση της αρεσκείας του. Με αυτόν τον τρόπο οι εφαρμογές αποκτούν το χαρακτηριστικό της εξατομίκευσης.

Στον κώδικα της βιβλιοθήκης υπάρχουν τμήματα τα οποία αποτελούσαν πρωτότυπα για την ανάπτυξη μιας νέας λειτουργίας που προσπάθησε να προστεθεί σε αυτήν από των προγραμματιστή της στην φάση ανάπτυξης της βιβλιοθήκης και είναι πιθανόν τα συγκεκριμένα τμήματα του κώδικα να μην είναι πλήρως λειτουργικά ή σε κάποιες περιπτώσεις είναι πιθανόν να μην έχει γίνει βελτιστοποίηση του κώδικα. Ενώ όπως διευκρινίζεται από τον σχεδιαστή και προγραμματιστή της βιβλιοθήκης στις σημειώσεις που συνοδεύουν τον κώδικα, η συγκεκριμένη προσπάθεια αποτελεί υλοποίηση προσωπικών ιδεών και εμπνεύσεων και είναι πιθανό κάποια πράγματα να μπορούσαν να υλοποιηθούν με πιο αποδοτικό τρόπο ή και να μπορούν να υλοποιηθούν και με άλλους τρόπους και μεθόδους. Τέλος, η προσπάθεια χαρακτηρίζεται ως μη-ιδανική και ημιτελής από τον ίδιο τον δημιουργό της.

1.6 Lidgren Network Library

Το Lidgren Network είναι μια δικτυακή βιβλιοθήκη του .Net Framework η οποία χρησιμοποιεί ένα μόνο UDP (User Datagram Protocol) socket για την παράδοση ενός API[8] (**A**pplication **P**rogramming **I**nterface) προκειμένου να επιτευχθεί η σύνδεση ενός πελάτη (Client) σε έναν εξυπηρετητή (Server), καθώς και η αποστολή και το διάβασμα των μηνυμάτων μεταξύ τους. Αυτή τη στιγμή η βιβλιοθήκη βρίσκεται στην τρίτη έκδοση και ο κώδικας της είναι «ανοιχτός» προς το ενδιαφερόμενο κοινό. Λόγω της αποκλειστικής χρήσης του UDP πρωτοκόλλου και των πλεονεκτημάτων που παρέχει το συγκεκριμένο πρωτόκολλο η βιβλιοθήκη έχει ιδιαίτερη απήχηση στον προγραμματισμό παιχνιδιών και ειδικότερα παιχνιδιών με μεγάλο ρυθμό μετάδοσης δεδομένων. Παρ' όλα αυτά η βιβλιοθήκη υλοποιεί μια εκδοχή του UDP όπου συνδυάζει τα πλεονεκτήματα του UDP και του TCP πρωτοκόλλου και μπορεί να χρησιμοποιηθεί ακόμα και αντικαθιστώντας σε πολλές περιπτώσεις την χρήση του TCP πρωτοκόλλου.

Η βιβλιοθήκη είναι εύκολη και απλή στη χρήση της, ενώ και σε συνδυασμό με την βιβλιογραφία, τα παραδείγματα, τους οδηγούς και τα προγράμματα που υπάρχουν στο διαδίκτυο η χρήση της γίνεται ακόμα πιο εύκολη. Χρήστες που δεν έχουν ασχοληθεί καθόλου με τον δικτυακό προγραμματισμό, μπορούν πλέον να δημιουργήσουν απλές εφαρμογές ανταλλαγής μηνυμάτων ή μικρά και απλά παιχνίδια κονσόλας. Με την χρήση της συγκεκριμένης βιβλιοθήκης, όλη η διαδικασία ανάπτυξης των εφαρμογών και των παιχνιδιών γίνεται μια εύκολη και διασκεδαστική εμπειρία για τους νέους στο χώρο προγραμματιστές. Ενώ με την περαιτέρω χρήση και εξοικείωση των εργαλείων και των τεχνικών επικοινωνίας και ανταλλαγής «πακέτων» μεταξύ πελατών και εξυπηρετητών μπορούν να μεταβούν στην ανάπτυξη πιο σύνθετων και μεγαλύτερων εφαρμογών και παιχνιδιών.

Όλη η λειτουργία της βιβλιοθήκης βασίζεται στην ανταλλαγή μηνυμάτων μεταξύ των συνδέσεων. Τα μηνύματα χωρίζονται σε δυο κατηγορίες, τα μηνύματα της βιβλιοθήκης τα οποία περιέχουν πληροφορίες όπως για την επίτευξη μιας σύνδεσης ή

[8] **A**pplication **P**rogram **I**nterface : Καλούμε τη διεπαφή των προγραμματιστικών διαδικασιών που ένα λειτουργικό σύστημα, βιβλιοθήκη ή εφαρμογή παρέχει προκειμένου να επιτρέπει να γίνονται προς αυτό αιτήσεις από άλλα προγράμματα ή / και ανταλλαγή δεδομένων.

διαγνωστικά μηνύματα (προειδοποιήσεις, σφάλματα) ή όταν κάτι αναπάντεχο συμβαίνει και τα μηνύματα δεδομένων τα οποία μεταφέρουν δεδομένα από έναν απομακρυσμένο χρήστη (συνδεμένο ή μη συνδεμένο). Η βασική κλάση για την δημιουργία συνδέσεων, την λήψη, την αποστολή και διαχείριση των μηνυμάτων κάνει χρήση της τεχνολογίας Peer-to-Peer[9] (P2P), αν παρ' όλα αυτά θέλετε να δημιουργήσετε μια τοπολογία και αρχιτεκτονική Client/Server η βιβλιοθήκη παρέχει μια ειδική κλάση η οποία κληρονομεί από την βασική κλάση και την επεκτείνει προσθέτοντας βοηθητικές μεθόδους και ιδιότητες ενώ ορίζει και κάποιες προεπιλεγμένες τιμές στις μεταβλητές.

Η παρούσα βιβλιοθήκη υλοποιεί και παρέχει πέντε διαθέσιμες μεθόδους για την αποστολή και παράδοση των μηνυμάτων μεταξύ των συνδέσεων:

α) `Unreliable`: αυτή η μέθοδος είναι το κοινό UDP πρωτόκολλο. Με αυτή τη μέθοδο υπάρχει η πιθανότητα κάποια από τα μηνύματα να «χαθούν». Επίσης μπορεί κάποια από αυτά να ληφθούν περισσότερες από μια φορές, ενώ ακόμα μπορεί τα μηνύματα να μην ληφθούν με την σειρά με την οποία στάλθηκαν.

β) `UnreliableSequenced`: χρησιμοποιώντας αυτή την μέθοδο παράδοσης υπάρχει ακόμα πιθανότητα τα μηνύματα να χαθούν, όμως σε αυτήν την περίπτωση υπάρχει προστασία απέναντι στην εμφάνιση διπλότυπων μηνυμάτων. Στην περίπτωση που κάποιο μήνυμα φτάσει πριν από τα προηγούμενα του, τότε αυτό το μήνυμα αποβάλλεται εξασφαλίζοντας έτσι ότι δεν θα υπάρχουν «παλιότερα» δεδομένα από αυτά που έχουν ήδη ληφθεί.

γ) `ReliableUnordered`: με αυτήν την μέθοδο εξασφαλίζουμε ότι τελικά όλα τα μηνύματα θα φτάσουν. Όμως δεν εγγυάται την σειρά με την οποία θα φτάσουν τα μηνύματα, επομένως είναι πιθανό «νεότερα» μηνύματα να ληφθούν πριν από τα «παλιότερα» μηνύματα.

δ) `ReliableSequenced`: η συγκεκριμένη μέθοδος μοιάζει με την `Unreliable Sequenced` με την διαφορά ότι εγγυάται ότι κάποια μηνύματα θα παραδοθούν. Στην περίπτωση που σταλεί ένα μόνο μήνυμα θα ληφθεί κανονικά, όμως στην περίπτωση του σταλούν δυο μηνύματα πολύ γρήγορα, και αυτά τα μηνύματα αναδιαταχτούν κατά την διαδικασία της μετάδοσης, τότε μόνο το «νεότερο» μήνυμα θα ληφθεί, όμως τουλάχιστον ένα από τα δυο μηνύματα θα ληφθεί σίγουρα.

ε) `ReliableOrdered`: η μέθοδος αυτή διασφαλίζει ότι τα μηνύματα θα παραλαμβάνονται πάντα και με την ίδια σειρά με την οποία στάλθηκαν.

Τα δεδομένα των εισερχόμενων μηνυμάτων διαχωρίζονται στους παρακάτω τύπους ανάλογα με τον σκοπό τον οποίο εξυπηρετούν (οι τύποι μηνυμάτων μπορούν

[9] Peer-to- Peer (ή **P2P**): Είναι ένα δίκτυο που επιτρέπει σε δύο ή περισσότερους υπολογιστές να μοιράζονται τους πόρους τους ισοδύναμα. Το δίκτυο αυτό χρησιμοποιεί την επεξεργαστική ισχύ, τον αποθηκευτικά χώρο και το εύρος ζώνης (bandwidth) των κόμβων. Όλοι οι κόμβοι του δικτύου έχουν ίσα δικαιώματα. Πληροφορίες που βρίσκονται στον ένα κόμβο, ανάλογα με τα δικαιώματα που καθορίζονται, μπορούν να διαβαστούν από όλους τους άλλους και αντίστροφα.

να ενεργοποιηθούν ή απενεργοποιηθούν χρησιμοποιώντας τις μεθόδους `EnableMessageType` και `DisableMessageType`):

- `Error`: αυτό το είδος μηνύματος υποδηλώνει ένα κατακερματισμένο μήνυμα και δεν πρέπει ποτέ να εμφανίζεται.

- `StatusChanged`: περιέχει ένα μόνο byte το οποίο μπορεί να μετασχηματιστεί στον τύπο απαρίθμησης `NetConnectionStatus` το οποίο περιέχει ένα αναγνώσιμο και επεξηγηματικό κείμενο για την αλλαγή κατάστασης. Λαμβάνονται μόνο μηνύματα που αφορούν την αλλαγή κατάστασης της συγκεκριμένης σύνδεσης και όχι για άλλες συνδέσεις που αφορούν τον ίδιο εξυπηρετητή στην περίπτωση της Client-Server αρχιτεκτονικής.

- `Data`: περιλαμβάνει τα δεδομένα τα οποία είναι γραμμένα από έναν συνδεδεμένο αποστολέα. Τα στοιχεία του αποστολέα είναι διαθέσιμα στην `SenderConnection` ιδιότητα του μηνύματος.

- `Unconnected Data`: σε αυτό το μήνυμα υπάρχουν δεδομένα γραμμένα από έναν χρήστη, ο οποίος δεν είναι συνδεμένος, αυτό το είδος των μηνυμάτων είναι απενεργοποιημένο ως προεπιλογή. Η `IP` διεύθυνση του αποστολέα είναι διαθέσιμη στην `SenderEndpoint` ιδιότητα του μηνύματος.

- `ConnectionApproval`: περιέχει τα δεδομένα που περάστηκαν κατά την διαδικασία έναρξης της σύνδεσης στην πλευρά του απομακρυσμένου χρήστη. Η εφαρμογή θα πρέπει να κάνει κλήση είτε της μεθόδου `Approve` ή της μεθόδου `Deny` στο αντικείμενο σύνδεσης της ιδιότητας `SenderConnection` του μηνύματος. Αυτός ο τύπος μηνύματος είναι απενεργοποιημένος ως προεπιλογή.

- `Receipt`: δεν έχει υλοποιηθεί ακόμα.

- `DiscoveryRequest`: είναι η διαδικασία που εκτελείται από τον πελάτη για τον εντοπισμό των διαθέσιμων εξυπηρετητών. Ο εντοπισμός μπορεί να γίνει με δυο τρόπους, είτε τοπικά μέσω της τεχνικής εκπομπής `broadcast`, η οποία θα στείλει σήμα σε όλες τις συνδέσεις του υποδικτύου ή απ' ευθείας επικοινωνία με μια `IP` διεύθυνση και ρωτώντας την αν κάποιος εξυπηρετητής είναι διαθέσιμος σε αυτή τη διεύθυνση.

- `DiscoveryResponse`: περιέχει δεδομένα για την απάντηση σε ένα `DiscoveryRequest`. Η διαδικασία απάντησης είναι ίδια ανεξάρτητα από τον τρόπο με τον οποίο γίνεται η αίτηση.

- `VerboseDebugMessage`: αποτελείται από ένα διαγνωστικό κείμενο. Είναι διαθέσιμο μόνο σε εκδόσεις απασφαλμάτωσης `DEBUG builds` και είναι απενεργοποιημένο ως προεπιλογή.

- `Debug Message`: περιέχει ένα διαγνωστικό κείμενο. Είναι διαθέσιμο μόνο σε εκδόσεις απασφαλμάτωσης `DEBUG builds`.

- `WarningMessage`: περιέχει ένα διαγνωστικό κείμενο.

- `ErrorMessage`: περιέχει ένα διαγνωστικό κείμενο.

- `NatIntroductionSuccess`: ενημέρωση για την επίτευξη επικοινωνίας μέσω της `NAT traversal` τεχνικής.

- `ConnectionLatencyUpdated`: ενημέρωση αλλαγής του μέσου όρου του χρόνου καθυστέρησης.

Κατά την μεταφορά των δεδομένων στο διαδίκτυο οι συνθήκες που επικρατούν είναι πιθανό να δημιουργήσουν όλων των ειδών τα προβλήματα. Τα πακέτα είναι πιθανόν να καθυστερήσουν, να χαθούν ή και να φτάσουν πάρα πολλές φορές στον προορισμό τους. Η βιβλιοθήκη `Lidgren network` υλοποιεί κάποιες λίγες τεχνικές για να μπορούν οι χρήστες της να προσομοιώνουν αυτές τις συνθήκες και να μπορούν να ελέγχουν την συμπεριφορά των εφαρμογών και των παιχνιδιών που υλοποιούν, όταν αυτές οι συνθήκες παρουσιάζονται. Στην κλάση `NetPeerConfiguration` μπορούν να ρυθμιστούν οι παρακάτω ιδιότητες:

α) `SimulatedLoss`: Μεταβλητή κινητής υποδιαστολής (Float) η οποία προσομοιώνει την απώλεια των πακέτων. Το εύρος τιμών της μεταβλητής είναι μεταξύ του μηδέν και του ένα, όπου το μηδέν αντιστοιχεί σε μηδενική απώλεια πακέτων και ουσιαστικά απενεργοποιεί την επιλογή. Επίσης όταν η τιμή είναι ίση με το ένα, τότε κανένα από τα απεσταλμένα πακέτα δεν θα φτάσει στον προορισμό του. Πρέπει να σημειωθεί ότι τα πακέτα μπορεί να περιέχουν αρκετά μηνύματα και ότι η «εξαφάνιση» των πακέτων γίνεται με τυχαία επιλογή.

β) `SimulatedDuplicatesChance`: Η ιδιότητα είναι μια μεταβλητή κινητής υποδιαστολής (Float) η οποία ορίζει το ποσοστό των πακέτων που θα διπλασιαστούν στον προορισμό τους. Το εύρος τιμών της μεταβλητής είναι μεταξύ του μηδέν και του ένα. Όπου στην περίπτωση του μηδέν κανένα πακέτο δεν θα διπλασιαστεί, ενώ στην περίπτωση που η τιμή είναι ίση με το ένα, τότε όλα τα πακέτα θα εμφανιστούν διπλότυπα στον προορισμό τους.

γ) `SimulatedMinimumLatency`: Αυτή η ιδιότητα προσομοιώνει την καθυστέρηση των πακέτων σε δευτερόλεπτα. Η τεχνητή καθυστέρηση προστίθεται στην πραγματική καθυστέρηση του δικτύου, έτσι η συνολική καθυστέρηση είναι η πραγματική

καθυστέρηση προς την μια κατεύθυνση προσθέτοντας την καθυστέρηση `SimulatedMinimumLatency` και την τυχαία καθυστέρηση `SimulatedRandomLatency`.

δ) `SimulatedRandomLatency`: Αυτή η ιδιότητα προσομοιώνει την τυχαία καθυστέρηση των πακέτων σε δευτερόλεπτα. Η τυχαία καθυστέρηση προστίθεται στην πραγματική καθυστέρηση του δικτύου, έτσι η συνολική καθυστέρηση είναι η πραγματική καθυστέρηση προς την μια κατεύθυνση προσθέτοντας την καθυστέρηση `SimulatedMinimumLatency` και την τυχαία καθυστέρηση `SimulatedRandomLatency`.

Για την καλύτερη και ευκολότερη χρήση των παραπάνω λειτουργιών συνιστάται η ανάθεση συμμετρικών συνθηκών και η ρύθμιση των εξυπηρετητών και των πελατών με τις ίδιες ρυθμίσεις. Η προσομοίωση «κακών» συνθηκών δικτύου λειτουργεί μόνο σε εκδόσεις απασφαλμάτωσης.

Κανονικά όταν γίνεται χρήση της συγκεκριμένης βιβλιοθήκης, η καλύτερη λύση για την ανταλλαγή μηνυμάτων είναι η δημιουργία μιας σύνδεσης και έπειτα η αποστολή των μηνυμάτων. Παρ' όλα αυτά η βιβλιοθήκη παρέχει την δυνατότητα στους χρήστες της για την αποστολή μηνυμάτων χωρίς την ύπαρξη σύνδεσης, το μόνο που χρειάζεται είναι τα στοιχεία (`IP` διεύθυνση, πόρτα) του χρήστη που θέλουμε να επικοινωνήσουμε, με την προϋπόθεση βέβαια πως ο χρήστης που θα δεχτεί το μήνυμα είναι ενεργός και έχει ενεργοποιήσει τον τύπο εισερχόμενων μηνυμάτων `UnconnectedData`.

Η βιβλιοθήκη υπερκαλύπτει `(override)` την μέθοδο `SendMessage` που χρησιμοποιείται για την αποστολή των μηνυμάτων, υλοποιώντας μια εκδοχή η οποία δέχεται έναν ακέραιο που ονομάζεται `sequenceChannel`, αυτή η μέθοδος χρησιμοποιείται με τις συγκεκριμένες μεθόδους αποστολής `UnreliableSequenced`, `ReliableSequenced` και `ReliableOrdered`. Το κανάλι ακολουθίας είναι ένας αριθμός μεταξύ του μηδέν και του τριάντα ένα (αυτήν τη στιγμή). Ο λόγος αυτού του περιορισμού είναι η προσπάθεια μείωσης του όγκου για κάθε μήνυμα. Να σημειωθεί ότι αυτός ο αριθμός των καναλιών είναι για κάθε μέθοδο αποστολής και όχι ο συνολικός αριθμός των καναλιών. Τα μηνύματα που θα παραδίδονται σε ένα συγκεκριμένο κανάλι ακολουθίας θα απορρίπτονται / παρακρατούνται ανεξάρτητα από τα μηνύματα που θα αποστέλλονται σε διαφορετικά κανάλια.

Η βιβλιοθήκη υλοποιεί και παρέχει στους χρήστες της μια σειρά από αλγορίθμους όπως τους `Xtea`, `AES`, `DES`, `Triple DES`, `RC2` και του απλού αλγορίθμου `XOR` για την κρυπτογράφηση των δεδομένων. Περά από τους υπάρχοντες αλγορίθμους κρυπτογράφησης οι χρήστες μπορούν να υλοποιήσουν τους δικούς τους αλγορίθμους. Η προσθήκη αλγορίθμων μπορεί να γίνει απλά χρησιμοποιώντας την βασική κλάση (`Base Class`) της βιβλιοθήκης που έχει υλοποιηθεί για την κρυπτογράφηση των δεδομένων. Έτσι κάθε κλάση που κληρονομεί από την κλάση `INetEncryption` και υλοποιεί τις μεθόδους της, μπορεί να χρησιμοποιηθεί ως τεχνική κρυπτογράφησης. Για αλγορίθμους τμηματικής κρυπτογράφησης (Block Cipher) υπάρχει η βασική κλάση `NetBlockEncryption` που μπορεί να χρησιμοποιηθεί για να απλοποιήσει την υλοποίηση τέτοιων αλγορίθμων κρυπτογράφησης, το μόνο που

χρειάζεται από τον χρήστη είναι η υλοποίηση των μεθόδων `EncryptBlock`, `DecryptBlock` και `BlockSize`. Η διαδικασία της κωδικοποίησης εφαρμόζεται εφόσον έχει γίνει η σύνταξη του μηνύματος και δεν πρέπει να προστεθούν στοιχεία σε αυτό μετά την εφαρμογή του αλγορίθμου, ενώ η αποκρυπτογράφηση είναι η πρώτη διαδικασία που γίνεται μετά την λήψη του μηνύματος. Για την διαδικασία της κρυπτογράφησης και της αποκρυπτογράφησης χρησιμοποιείται ένας κωδικός – κλειδί. Για τον συγχρονισμό και τον διαμοιρασμό του κλειδιού δεν έχει υλοποιηθεί κάποια τεχνική. Έτσι για την γνωστοποίηση του κλειδιού στους παραλήπτες θα πρέπει να υπάρξει ιδιαίτερη μέριμνα από τους χρήστες της βιβλιοθήκης. Το στιγμιότυπο της κλάσης του αλγορίθμου μπορεί να χρησιμοποιηθεί για όλη την διάρκεια της «ζωής» του, όμως η κλάση δεν είναι ασφαλής για χρήση σε περισσότερα του ενός νήματος. Επομένως θα πρέπει να δημιουργείται ένα στιγμιότυπο για κάθε νήμα στο οποίο γίνεται κρυπτογράφηση ή αποκρυπτογράφηση.

Υπάρχουν τρεις τρόποι που χρησιμοποιούνται για την ανάγνωση των μηνυμάτων που λαμβάνονται από την βιβλιοθήκη.

1. `Polling`: εάν υπάρχει ήδη ένας βρόχος στην εφαρμογή αυτός είναι ο καταλληλότερος τρόπος, απλά γίνεται ανάγνωση των μηνυμάτων μέσα στον βρόγχο με την χρήση της μεθόδου `ReadMessage` μέχρι να επιστρέψει κενό περιεχόμενο (`null`).

2. `Callback`: εάν η εφαρμογή έχει υλοποιηθεί με την τεχνική του προγραμματισμού χειρισμού γεγονότων (Event - Driven Programming) ή του γραφικού περιβάλλοντος χρήστη (Graphical User Interface) και δεν υπάρχει βρόχος επανάληψης στην εφαρμογή, τότε μπορεί να γίνει εγγραφή μιας μεθόδου επιστροφής κλήσης η οποία θα «πυροδοτηθεί» όταν φτάσει ένα μήνυμα.

3. `EventWaitHandle`: εάν η διαδικασία επεξεργασίας του εισερχόμενου μηνύματος «τρέχει» σε ένα ξεχωριστό νήμα, μπορεί να γίνει χρήση της ιδιότητας `MessageReceivedEvent` προκειμένου να μπλοκάρει το νήμα έως ότου ένα εισερχόμενο μήνυμα ληφθεί.

-Κεφάλαιο 2ο-

Απαιτήσεις Λογισμικού

2. Απαιτήσεις Λογισμικού

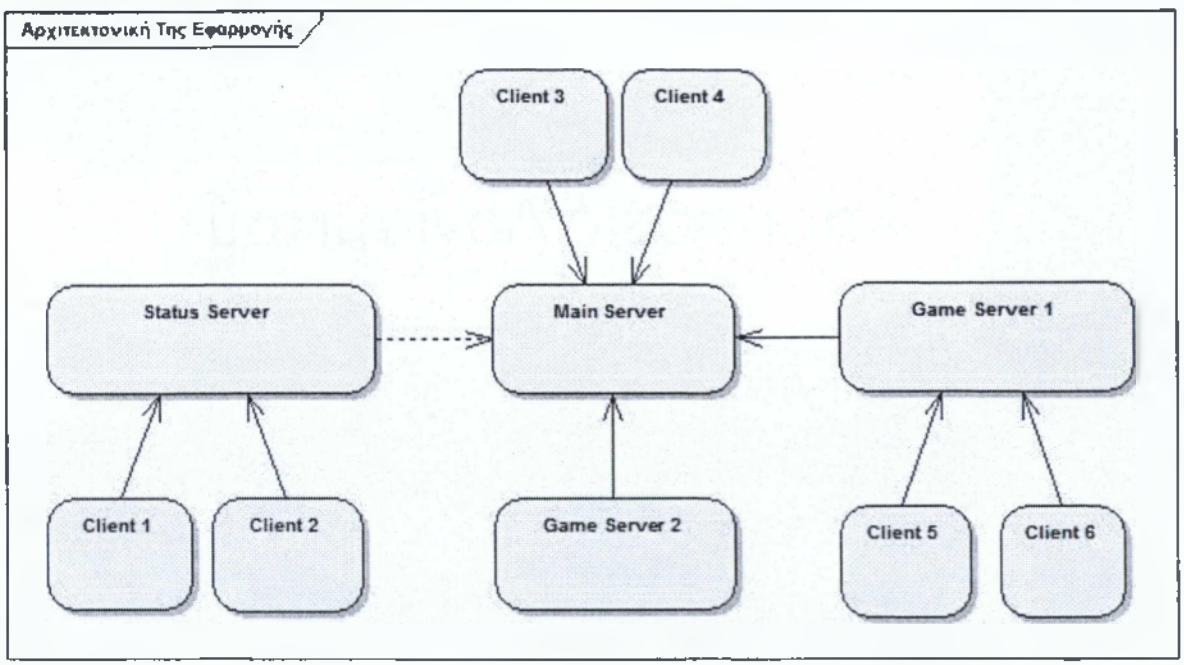
Οι ενότητες οι οποίες απαρτίζουν το παρόν κεφάλαιο περιγράφονται συνοπτικά παρακάτω:

Η ενότητα «Αρχιτεκτονική της εφαρμογής» παρουσιάζει τη δομή της αρχιτεκτονικής της εφαρμογής.

Οι ενότητες «Λειτουργικές Απαιτήσεις Εξυπηρετητή» και «Λειτουργικές απαιτήσεις Πελάτη» περιέχουν μια επισκόπηση όλων των λειτουργιών, της συμπεριφοράς, των επιχειρησιακών κανόνων, του γραφικού περιβάλλοντος χρήστη και της γενικής λειτουργικότητας του λογισμικού το οποίο πρόκειται να αναπτυχθεί.

Η ενότητα «Μη Λειτουργικές Απαιτήσεις» περιέχει μία περιγραφή των απαιτήσεων του συστήματος, καθώς και λεπτομερειών που αφορούν την απόδοση, ασφάλεια, επεκτασιμότητα, επιμονή, μεταφορά και τη δυνατότητα συντήρησης τις οποίες το προτεινόμενο σύστημα θα υποστηρίζει.

2.1 Αρχιτεκτονική της εφαρμογής



Σχήμα 2.1-1: Αρχιτεκτονική της εφαρμογής

Για την ευκολότερη ανάπτυξη, συντήρηση και αποσφαλμάτωση των εξυπηρετητών έγινε κατάτμηση τους σε τρεις μικρότερους συμφώνα με τις λειτουργιές που εκτελούν. Αρχικά θέλαμε να υπάρχει ένας εξυπηρετητής ο οποίος θα είναι σε συνεχή «εκτέλεση» και σκοπός του θα είναι να ενημερώνει τους χρήστες για την κατάσταση των υπολοίπων εξυπηρετητών. Στην παρούσα φάση ο συγκεκριμένος διακομιστής «διαβάζει» την κατάσταση και την έκδοση της εφαρμογής πελάτη από ένα αρχείο και μεταβιβάζει αυτές τις πληροφορίες στους πελάτες που προσπαθούν να

συνδεθούν. Κατά την διάρκεια των δοκιμών (Beta Tests) επιπρόσθετος σκοπός του διακομιστή θα είναι να αποτελέσει συνδετικό κρίκο μεταξύ των διαχειριστών του συστήματος και των χρηστών που συμμετέχουν στην διαδικασία των δοκιμών (Beta Tests). Στο πλαίσιο αυτής της πτυχιακής δεν θα υπάρξει διαδικασία δοκιμών.

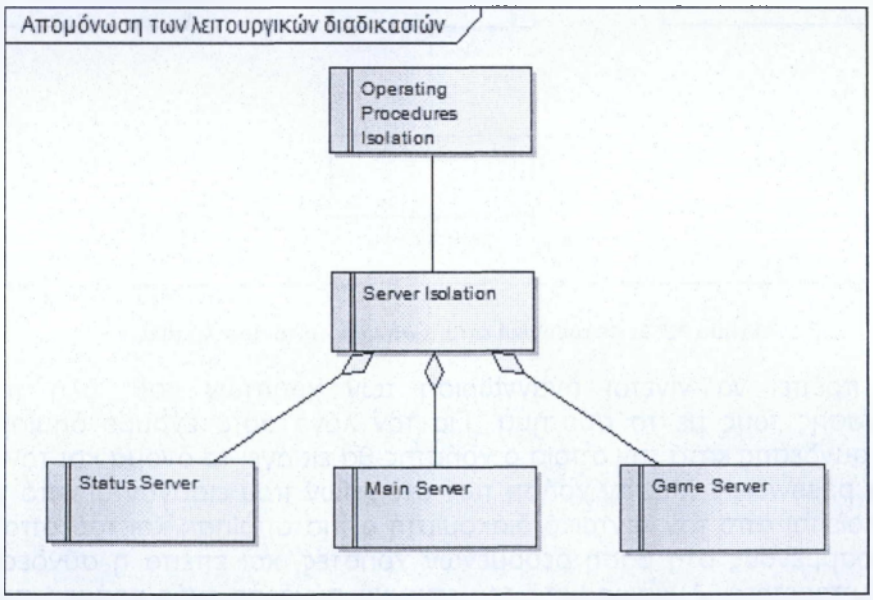
Ο κεντρικός εξυπηρετητής έχει ως σκοπό την διαχείριση της σύνδεσης των χρηστών και των εξυπηρετητών παιχνιδιού που θα υλοποιήσουν την λογική του παιχνιδιού. Επιπλέον θα συγκεντρώνει πληροφορίες για τα υπάρχοντα τραπέζια και θα μεταφέρει τις πληροφορίες στους πελάτες. Τέλος μια από κύριες λειτουργίες του εξυπηρετητή είναι η διαχείριση των μηνυμάτων που θα ανταλλάσσονται μεταξύ των πελατών ενώ και οι διαχειριστές θα έχουν την δυνατότητα να επικοινωνούν με τους χρήστες μέσω μηνυμάτων τα οποία θα έχουν ειδική σήμανση.

Οι εξυπηρετητές παιχνιδιού έχουν σαν σκοπό την υλοποίηση της λογικής του παιχνιδιού. Σε αυτούς συνδέονται οι χρήστες προκειμένου να παρακολουθήσουν ή να παίξουν στο τραπέζι που «φιλοξενεί» ο εξυπηρετητής παιχνιδιού. Κάθε εξυπηρετητής θα «φιλοξενεί» μόνο ένα τραπέζι προς το παρόν, ενώ υπάρχει και περιορισμός στους εξυπηρετητές παιχνιδιού που εκτελούνται πάνω σε μια δημόσια ηλεκτρονική διεύθυνση (public IP) για λόγους απόδοσης.

Οι Λειτουργικές Απαιτήσεις διακρίνονται σε δύο κατηγορίες: α) Στις Λειτουργικές απαιτήσεις εξυπηρετητή οι οποίες αφορούν στις απαιτήσεις της κεντρικής εφαρμογής που υλοποιούνται από την πλευρά των εξυπηρετητών. β) Στις Λειτουργικές απαιτήσεις πελάτη οι οποίες αφορούν στις απαιτήσεις του συστήματος τις οποίες υλοποιήσαμε στην εφαρμογή πελάτη (Client).

2.2 Λειτουργικές Απαιτήσεις Εξυπηρετητή

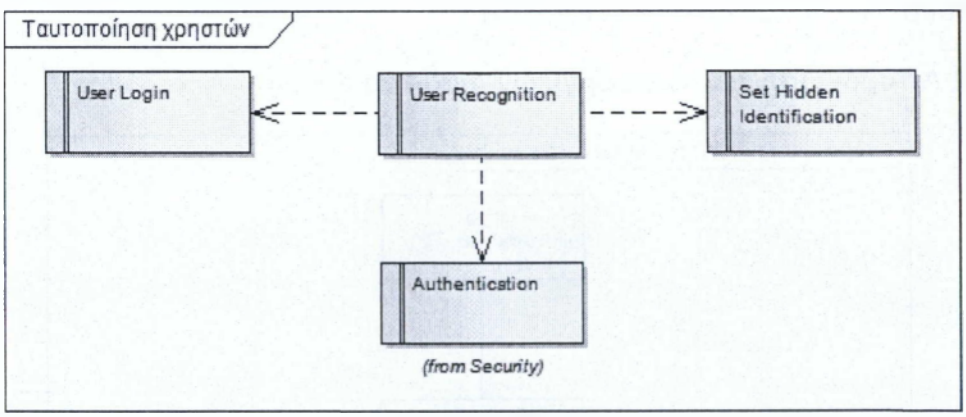
2.2.1 Απομόνωση των λειτουργικών διαδικασιών



Σχήμα 2.2-5: Χαρακτηριστικό «Απομόνωση Λειτουργικών Διαδικασιών»

Για την καλύτερη διαχείριση των πόρων, την ευκολότερη ανάπτυξη και συντήρηση των εξυπηρετητών καθώς και την απομόνωση των σφαλμάτων που μπορεί να προκύψουν στον εξυπηρετητή κατά την εκτέλεση του έγινε διαχωρισμός τους συμφωνά με τις διακριτές διεργασίες που εκτελούν. Έτσι εξαλείφεται η ανάγκη χρήσης μεγάλων σε υπολογιστική ισχύ συστημάτων για την εκτέλεση και υποστήριξη του «τεράστιου» εξυπηρετητή, έχοντας παράλληλα και την δυνατότητα να «τρέχουμε» τους διακομιστές σε ανεξάρτητα μεταξύ τους συστήματα και κανάλια επικοινωνίας. Με αυτόν τον τρόπο υπάρχει η δυνατότητα της εκτέλεσης των εφαρμογών των διακομιστών από καθημερινούς προσωπικούς υπολογιστές και με την χρήση των κοινών συνδέσεων διαδικτύου. Για την επίτευξη της παραπάνω λειτουργίας υλοποιήσαμε τρεις διαφορετικούς εξυπηρετητές καθ' ένας εκ των οποίων έχει τον δικό του διαφορετικό ρόλο και μαζί συνθέτουν την τελική εφαρμογή. Έτσι υπάρχει ο διακομιστής κατάστασης που ενημερώνει τους χρήστες για την κατάσταση των υπόλοιπων εξυπηρετητών, ο κεντρικός εξυπηρετητής που διαχειρίζεται τους χρήστες και τους διακομιστές μετά την σύνδεση τους στο σύστημα και ο διακομιστής παιχνιδιού στον οποίο πραγματοποιείται η εκτέλεση της διαδικασίας και των κανόνων του πόκερ. Κύριο χαρακτηριστικό του τελευταίου είναι η ικανότητα να εκτελείται σε ανεξάρτητο και διαφορετικό σύστημα από τους υπόλοιπους εξυπηρετητές, ενώ για κάθε τραπέζι «εκτελείται» και διαφορετικός διακομιστής πετυχαίνοντας έτσι την απομόνωση μεταξύ των εξυπηρετητών για λόγους ασφαλείας και απόδοσης αυτών.

2.2.2 Ταυτοποίηση χρηστών από τον Κεντρικό Εξυπηρετητή

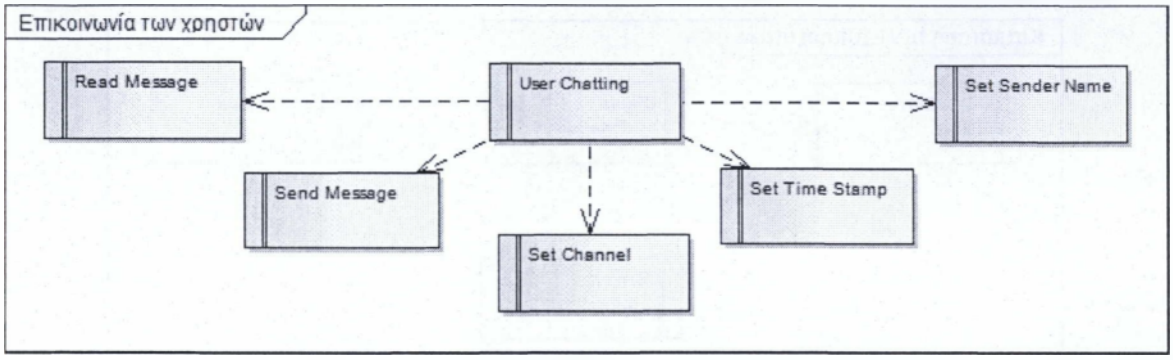


Σχήμα 2.2-2: Λειτουργική απαίτηση «Ταυτοποίηση Χρηστών»

Θα πρέπει να γίνεται αναγνώριση των χρηστών καθ' όλη την διάρκεια αλληλεπίδρασης τους με το σύστημα. Για τον λόγο αυτό έχουμε δημιουργήσει την διαδικασία σύνδεσης κατά την οποία ο χρήστης θα εισάγει το όνομα και τον κωδικό του (username, password). Με την χρήση των στοιχείων που εισάγονται από τον χρήστη, πραγματοποιείται από τον κεντρικό διακομιστή η πιστοποίηση και ταυτοποίηση του με τους εγγεγραμμένους στη βάση δεδομένων χρήστες, και έπειτα η σύνδεση του στον κεντρικό εξυπηρετητή. Αμέσως μετά την επιτυχή σύνδεση ενός χρήστη στο σύστημα, παρέχεται σε αυτόν ένα μοναδικό αναγνωριστικό το οποίο θα χρησιμοποιείται για την

μετέπειτα αναγνώριση του. Έτσι κάθε φορά κατά την οποία κάποιος χρήστης θέλει να επικοινωνήσει με τους διακομιστές θα στέλνει σε αυτούς το μοναδικό αναγνωριστικό το οποίο είναι γνωστό μόνο από την συγκεκριμένη εφαρμογή πελάτη και χρησιμοποιείται για την συνεχή ταυτοποίηση του.

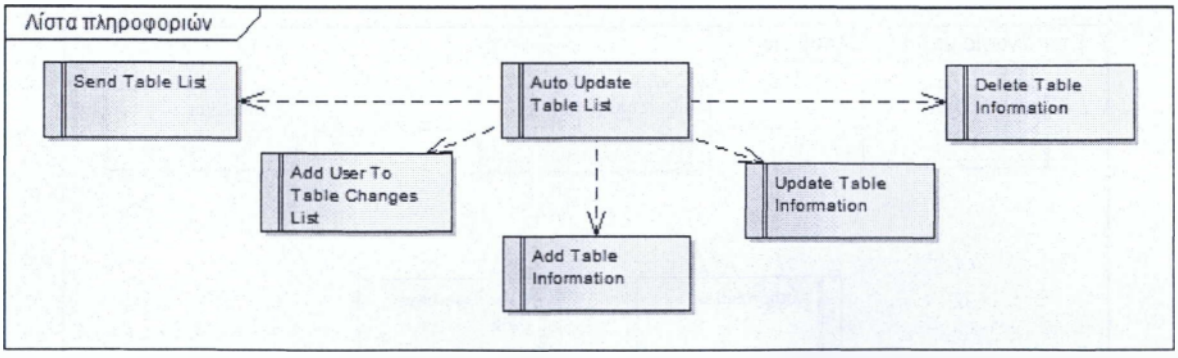
2.2.3 Επικοινωνία μεταξύ των χρηστών



Σχήμα 2.2-3: Λειτουργική απαίτηση «Επικοινωνία Χρηστών»

Το σύστημα θα πρέπει να παρέχει στους συνδεμένους χρήστες του την δυνατότητα της επικοινωνίας με την χρήση άμεσων μηνυμάτων. Τα μηνύματα θα τοποθετούνται σε «κανάλια» ανάλογα με τον αποστολέα και θα έχουν ειδική σήμανση στην περίπτωση που θα συντάσσονται από διαχειριστές του συστήματος. Επίσης στα μηνύματα θα πρέπει να υπάρχει χρονική σήμανση σε ορισμένες περιπτώσεις. Η διαδικασία που ακολουθείται για την παράδοση των μηνυμάτων απαιτεί την συμμετοχή του κεντρικού διακομιστή ο οποίος δέχεται εισερχόμενα μηνύματα από τους χρήστες και τα μεταβιβάζει στους υπόλοιπους συνδεδεμένους χρήστες.

2.2.4 Λίστα πληροφοριών αυτόματης ανανέωσης

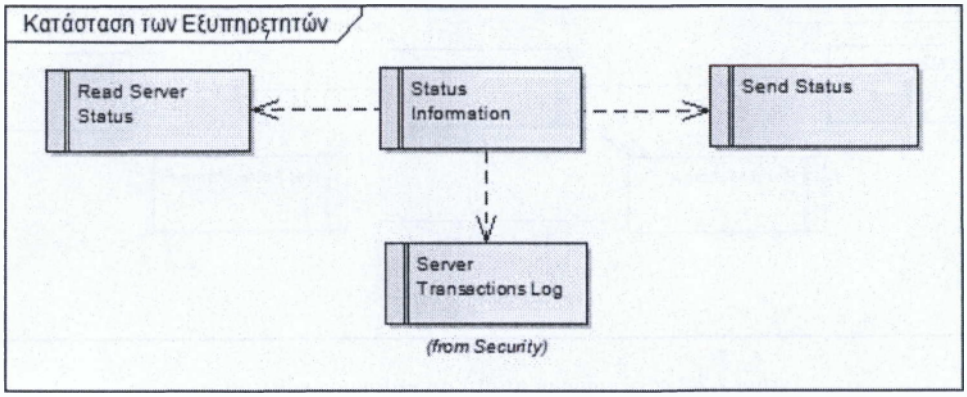


Σχήμα 2.2-4: Λειτουργική απαίτηση «Λίστα Πληροφοριών»

Ο κεντρικός εξυπηρετητής θα διαθέτει λίστα με πληροφορίες για τα «ενεργά» τραπέζια, καθώς και τις βασικές πληροφορίες του κάθε τραπεζιού, οι πληροφορίες της

λίστας γνωστοποιούνται στους χρήστες αμέσως μετά την σύνδεση τους στον κεντρικό εξυπηρετητή ενώ για την μετέπειτα ενημέρωση τους ο κεντρικός εξυπηρετητής ταυτόχρονα θα «εγγράφει» τους χρήστες σε μία λίστα για την αυτόματη ενημέρωση τους σχετικά με τις μεταβολές (έναρξη, ενημέρωση και τερματισμό) των τραπεζιών.

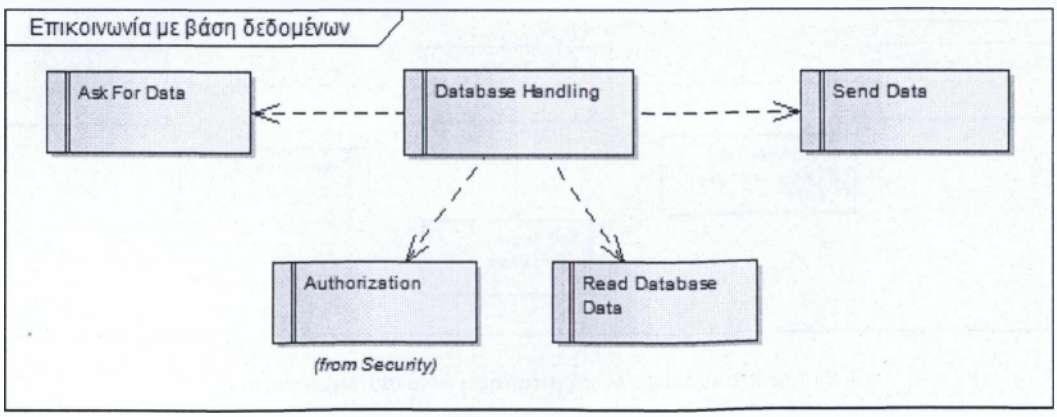
2.2.5 Κατάσταση των εξυπηρετητών



Σχήμα 2.2-1: Λειτουργική απαίτηση «Κατάστασης Εξυπηρετητών»

Κατά την εκκίνηση της εφαρμογής πελάτη θέλουμε αυτή να ενημερώνεται για την κατάσταση των εξυπηρετητών και να εμφανίζει στον χρήστη πληροφορίες για την κατάσταση στην οποία βρίσκονται πριν την διαδικασία της πιστοποίησης – σύνδεσης των χρηστών. Για την επίτευξη της παραπάνω πρότασης υλοποιήσαμε έναν αυτόνομο διακομιστή, σκοπός του οποίου είναι να ενημερώνεται για την κατάσταση των υπόλοιπων εξυπηρετητών και να μεταβιβάζει τις πληροφορίες στους συνδεδεμένους σε αυτόν πελάτες.

2.2.6 Επικοινωνία με βάση δεδομένων



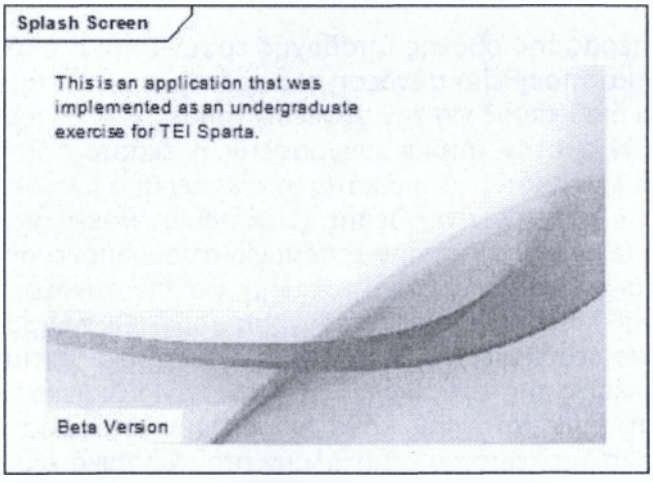
Σχήμα 2.2-6: Λειτουργική απαίτηση «Επικοινωνία με Βάση Δεδομένων»

Στο σύστημα στο οποίο εκτελείται ο κεντρικός εξυπηρετητής πρέπει να υπάρχει εγκατεστημένη βάση δεδομένων, για την διαχείριση των αποθηκευμένων πληροφοριών που απαιτούνται κατά την διάρκεια χρήσης της εφαρμογής από τους χρήστες της, καθώς και για την δυνατότητα για αποθήκευση των δεδομένων που παράγονται κατά την διάρκεια εκτέλεσης των εξυπηρετητών. Τα δεδομένα της βάσης είναι προσπελάσιμα μόνο από τον κεντρικό διακομιστή. Έτσι όταν κάποιος εξυπηρετητής που εκτελεί την διαδικασία του παιχνιδιού χρειάζεται πρόσβαση στα δεδομένα της βάσης αιτείται από τον κεντρικό εξυπηρετητή ο οποίος μετά τον έλεγχο για τα δικαιώματα πρόσβασης στα δεδομένα «διαβάζει» τα δεδομένα από την βάση και του τα μεταβιβάζει κάθε φορά που τα χρειάζεται. Με αυτόν τον τρόπο οι διακομιστές που εκτελούν την διαδικασία του παιχνιδιού δεν απαιτούν την ύπαρξη βάσης δεδομένων και μπορούν να εκτελούνται σε συστήματα απομακρυσμένα από τον κεντρικό εξυπηρετητή.

2.3 Λειτουργικές απαιτήσεις Πελάτη

Στην ενότητα αυτή παρουσιάζονται οι λειτουργικές απαιτήσεις του συστήματος από την πλευρά της εφαρμογής πελάτη. Για την παρουσίαση των γραφικών απαιτήσεων της εφαρμογής χρησιμοποιήσαμε την δυνατότητα που μας παρέχει το πρόγραμμα σχεδίασης UML διαγραμμάτων για την σχεδίαση γραφικών παραθύρων. Έτσι σχεδιάσαμε τις τέσσερις διαφορετικές οθόνες που θα εμφανίζονται κατά την εκτέλεση της εφαρμογής.

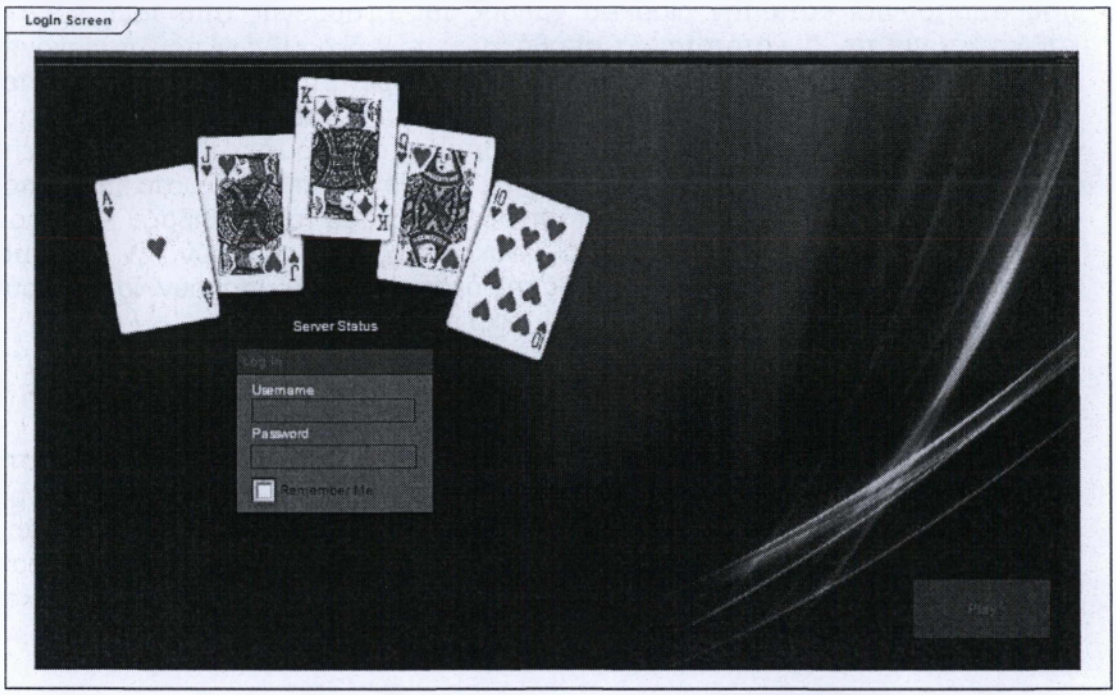
2.3.1 Οθόνη υποδοχής



Σχήμα 2.3-1: UML απεικόνιση οθόνης υποδοχής

Κατά την εκκίνηση της εφαρμογής εμφανίζεται στον χρήστη ένα παράθυρο το οποίο περιέχει πληροφορίες σχετικά με τον σκοπό και το πλαίσιο κάτω από το οποίο υλοποιήθηκε η εφαρμογή και με την έκδοση της εφαρμογής πελάτη που εκτελείται.

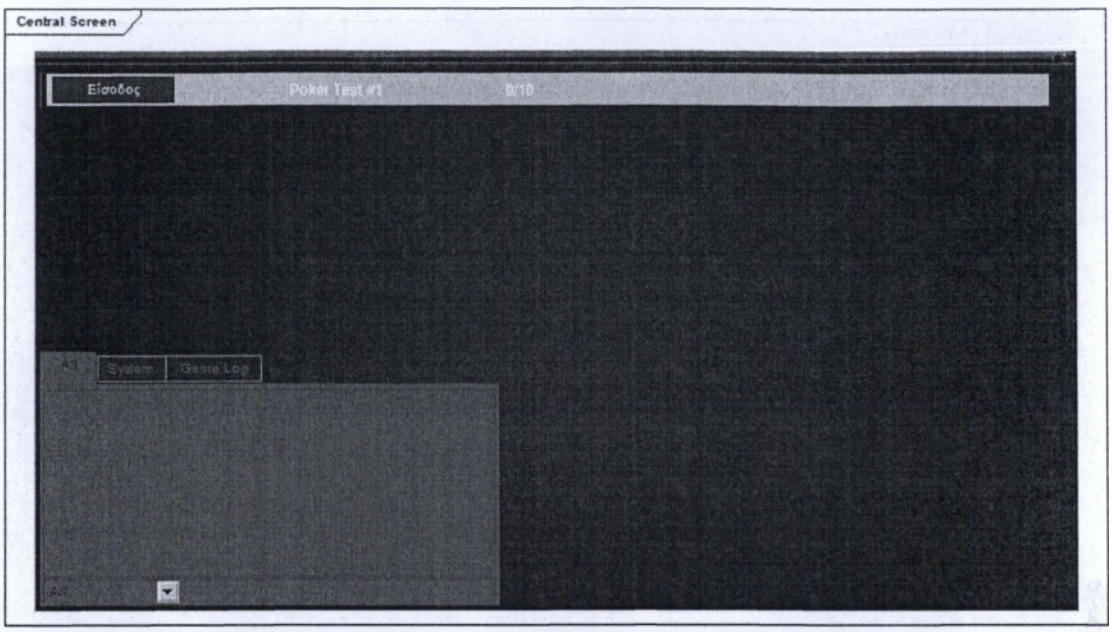
2.3.2 Οθόνη σύνδεσης



Σχήμα 2.3-2: UML απεικόνιση οθόνης σύνδεσης

Μετά το πέρας της οθόνης υποδοχής εμφανίζεται η οθόνη σύνδεσης μέσω της οποίας θα πραγματοποιηθεί η σύνδεση του χρήστη στο σύστημα εφ' όσον ο κεντρικός διακομιστής είναι διαθέσιμος για την σύνδεση χρηστών σε αυτόν. Στην οθόνη σύνδεσης υπάρχει μια λεζάντα στην οποία αναγράφεται η εκάστοτε κατάσταση του κεντρικού εξυπηρετητή ενώ λίγα εκατοστά πιο κάτω υπάρχουν δυο πλαίσια κειμένου (Textbox) για την εισαγωγή των στοιχείων σύνδεσης (Username, Password). Επίσης υπάρχει ένα πλαίσιο ελέγχου (Checkbox) για την λειτουργία απομνημόνευσης (Remember Me) των στοιχείων σύνδεσης που θα χρησιμοποιηθούν για την σύνδεση του χρήστη, έτσι ώστε να μην χρειάζεται η εκ νέου πληκτρολόγηση τους. Οι πληροφορίες που εισάγονται αποθηκεύονται σε ένα αρχείο ρυθμίσεων από το οποίο γίνεται η ανάγνωση τους σε μελλοντικές εκτελέσεις της εφαρμογής. Η παραπάνω διαδικασία μπορεί να ακυρωθεί από τον χρήστη πριν από την κάθε είσοδο του στο σύστημα. Τα στοιχεία που εισάγονται στα πλαίσια κειμένου θα σταλούν στον κεντρικό εξυπηρετητή είτε πιέζοντας το κουμπί σύνδεσης (Play) είτε με χρήση του πλήκτρου Enter από το πληκτρολόγιο του χρήστη εφ' όσον έχουν συμπληρωθεί τα πλαίσια κειμένου.

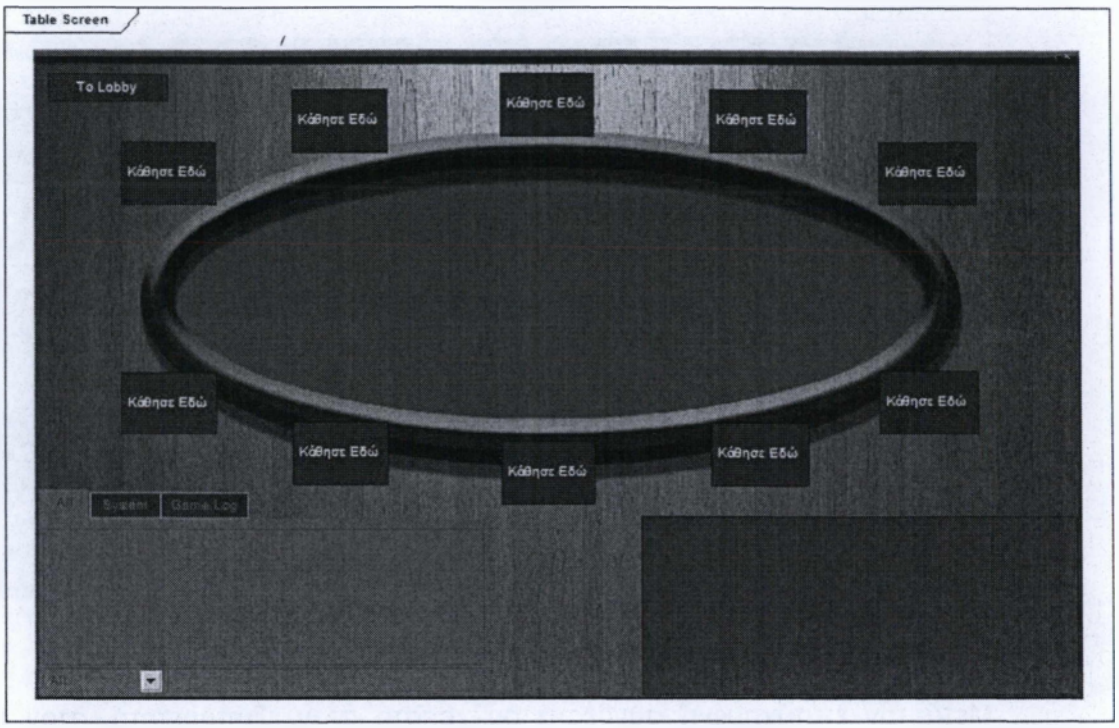
2.3.3 Κεντρική οθόνη



Σχήμα 2.3-3: UML απεικόνιση κεντρικής οθόνης

Μετά την επιτυχημένη σύνδεση του χρήστη στον εξυπηρετητή, στον πελάτη εμφανίζεται η κεντρική οθόνη της εφαρμογής. Καθ' όλη την παρουσία του σε αυτήν, υπάρχει στην κάτω αριστερή γωνία του παράθυρου ειδική διεπαφή για την πληκτρολόγηση μηνυμάτων, αποστολή στον διακομιστή και προώθηση τους από τον εξυπηρετητή. Ο τελευταίος είναι αυτός που έχει επωμιστεί την ευθύνη για την διαχείριση των μηνυμάτων από τους χρήστες της εφαρμογής και τη μετέπειτα εμφάνιση των μηνυμάτων σε αυτούς. Η διεπαφή διαθέτει καρτέλες (tabs) για την ομαδοποίηση των μηνυμάτων ανάλογα με τον αποστολέα τους. Κατά την εμφάνιση των μηνυμάτων προστίθεται χρονική σήμανση σε αυτά, ενώ σε ορισμένες περιπτώσεις εμφανίζεται και το όνομα του καναλιού στο οποίο ανήκει το μήνυμα. Στο επάνω μέρος του παράθυρου ένα μεγάλο μέρος του καταλαμβάνεται από έναν γραφικό πίνακα στον οποίο θα εμφανίζονται οι πληροφορίες (όνομα, αριθμός υπαρχόντων παικτών και μέγιστος αριθμός παικτών) των «ενεργών τραπεζιών». Για κάθε «τραπέζι» περιλαμβάνεται ένα κουμπί (button) μέσω του οποίου ο χρήστης έχει την δυνατότητα να μεταβεί από την κεντρική οθόνη στην οθόνη όπου εκτελούνται οι κανόνες του παιχνιδιού, προκειμένου είτε να παρακολουθήσει είτε να συμμετάσχει στις παρτίδες του παιχνιδιού. Η ανανέωση των πληροφοριών γίνεται αυτόματα και δεν απαιτείται κάποια ενέργεια από τον χρήστη.

2.3.4 Οθόνη «τραπεζιού»



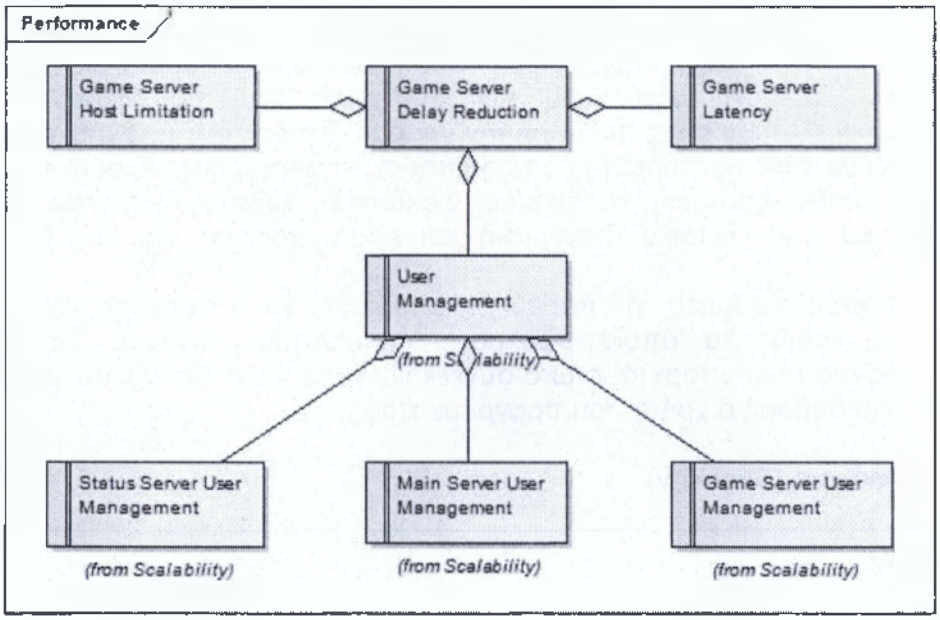
Σχήμα 2.3-4: UML απεικόνιση οθόνης τραπεζιού

Με την σύνδεση των χρηστών σε κάποιον εξυπηρετητή παιχνιδιού εμφανίζεται στην εφαρμογή του πελάτη η διεπαφή μέσω της οποίας θα μπορεί ο χρήστης να παρακολουθήσει και να συμμετάσχει στις παρτίδες του παιχνιδιού που επιθυμεί. Η εφαρμογή χρησιμοποίει δισδιάστατα γραφικά αρχείων εικόνας που αναπαριστούν ένα εικονικό τραπέζι, γύρω από το οποίο θα «συγκεντρώνονται» οι χρήστες για να παίξουν στις παρτίδες του παιχνιδιού. Για να μπορέσει ο χρήστης να συμμετάσχει στις παρτίδες του παιχνιδιού θα πρέπει πρωτίστως να λάβει μια θέση στο εικονικό τραπέζι. Αυτό γίνεται πολύ απλά «πιέζοντας» ένα από τα κουμπιά (Buttons) στο οποίο αναγράφεται η φράση «Κάθισε Εδώ». Έτσι ο χρήστης θα δεσμεύσει την θέση δίνοντας του το δικαίωμα να συμμετάσχει στην επόμενες παρτίδες του παιχνιδιού. Μόλις ο χρήστης «καθίσει» σε μια θέση στο εικονικό τραπέζι, εμφανίζεται επάνω αριστερά στην εφαρμογή του πελάτη ένα κουμπί το οποίο του επιτρέπει να «σηκωθεί» (Sit Out) από το τραπέζι όταν αυτός το επιθυμεί. Στην ίδια περιοχή βρίσκεται και ένα κουμπί (To Lobby) που επιτρέπει στους χρήστες να επιστέψουν στην κεντρική οθόνη. Στο κάτω μέρος και δεξιά της εφαρμογής βρίσκεται ένα πλαίσιο μέσα στο οποίο εμφανίζονται στοιχεία ελέγχου (Controls) ανάλογα με τις περιστάσεις. Στο κάτω και αριστερό μέρος της εφαρμογής βρίσκεται η ίδια διεπαφή που υπάρχει και στην κεντρική οθόνη και χρησιμοποιείται για την επικοινωνία μεταξύ των χρηστών.

2.4 Μη Λειτουργικές Απαιτήσεις

Λόγω της φύσης της εργασίας, αποφασίσαμε να μην επεκταθούμε περισσότερο σε θέματα που αφορούν στην αποδοτικότητα και την ασφάλεια της εφαρμογής μας παρά μόνο να καλύψουμε τα βασικά θέματα προκειμένου να μπορεί να «τρέχει» σε ένα απλοϊκό επίπεδο. Ειδικά όσον αφορά στην ασφάλεια ακολουθήσαμε την παραδοχή ότι πρόκειται για μια εφαρμογή με σκοπό την διασκέδαση. Θεωρήσαμε λοιπόν ότι δεν θα υπάρξουν «κακόβουλοι» χρήστες, οι όποιοι θα προσπαθήσουν με παράνομα μέσα και τεχνικές να βλάψουν την ομαλή λειτουργία του προγράμματος με σκοπό να επωφεληθούν από αυτό.

2.4.1 Απόδοση (Performance)



Σχήμα 2.4-1: Μη-λειτουργική απαίτηση «Απόδοση»

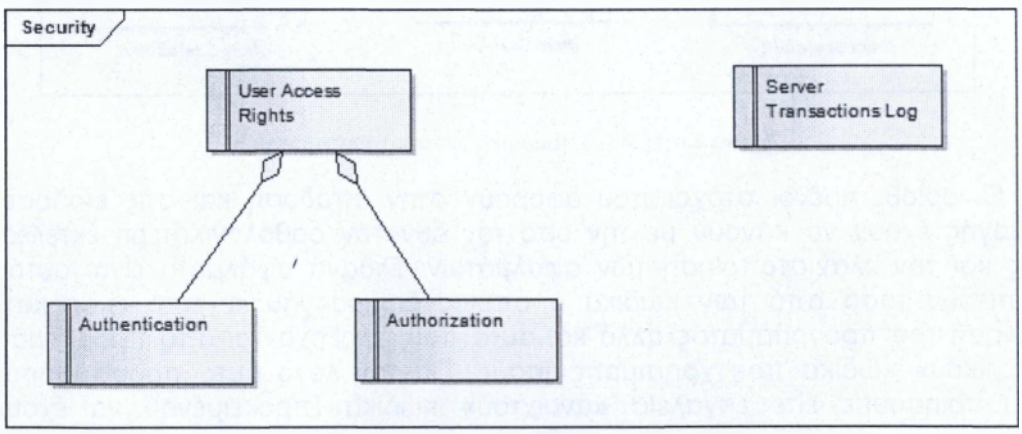
Οι οριοθετημένοι στόχοι που αφορούν στην απόδοση και στις επιδόσεις της εφαρμογής έχουν να κάνουν με την όσο τον δυνατόν ορθολογικότερη εκτέλεση της καθώς και την ελαχιστοποίηση των σφαλμάτων. Πιθανά σφάλματα είναι αυτά που προκύπτουν τόσο από των κώδικα ο όποιος δημιουργήθηκε από εμάς κατά την υλοποίηση του προγράμματος αλλά και αυτά που προέρχονται από προβλήματα του «εξωτερικού» κώδικα που χρησιμοποιήσαμε. Για τον λόγο αυτό προσπαθήσαμε να χρησιμοποιήσουμε είτε εργαλεία «ανοιχτού» κώδικα (προκειμένου να έχουμε τη δυνατότητα να επεμβαίνουμε στον κώδικα), είτε εργαλεία ευρέως διαδεδομένα στον χώρο της πληροφορικής. Για τα εργαλεία αυτά είχαμε τη δυνατότητα για αναζήτηση βοήθειας στο διαδίκτυο για την επίλυση πιθανών προβλημάτων, με βασική πάντα προϋπόθεση να μπορούμε να τα προμηθευτούμε χωρίς κάποιο κόστος.

Τα κυριότερα προβλήματα τα όποια εμφανιστήκαν και τα οποία είτε δεν επέτρεπαν την ορθή λειτουργία της εφαρμογής πελάτη είτε εμπόδιζαν την ευκολότερη χρήση του γραφικού περιβάλλοντος της από τον χρήστη αντιμετωπίστηκαν επιτυχώς, επεμβαίνοντας στον «ανοιχτό» κώδικα της εφαρμογής. Όσον αφορά τώρα στο εργαλείο που χρησιμοποιήθηκε για την επικοινωνία των πελατών με τους εξυπηρετητές και τα προβλήματα που προέκυψαν από αυτό, τα περισσότερα αντιμετωπίστηκαν με επιτυχία. Ενώ μόνο σε σπάνιες περιπτώσεις οι εξυπηρετητές «παγώνουν» και δεν καταφέρνουν να επικοινωνήσουν μεταξύ τους και θα πρέπει να γίνεται η επανεκκίνηση των εξυπηρετητών και των πελατών για να λειτουργήσουν ξανά σωστά από την αρχή.

Για την διασφάλιση της ποιότητας της επικοινωνίας των εξυπηρετητών με τους πελάτες, γίνεται έλεγχος για τον αριθμό των διακομιστών παιχνιδιού που «τρέχουν» πάνω σε μια γραμμή επικοινωνίας, θέτοντας ένα μέγιστο όριο για τους εξυπηρετητές που μπορούν να εκτελεστούν από την ίδια παγκόσμια ηλεκτρονική διεύθυνση σε κάθε περίπτωση, την δεδομένη στιγμή ο μέγιστος αριθμός των ταυτόχρονων εξυπηρετητών αντιστοιχεί σε δέκα. Ενώ σε δεύτερη φάση δεν επιτρέπουμε σε γραμμές με καθυστέρηση (Latency) μεγαλύτερη των 200 χιλιοστών του δευτερόλεπτου να «φιλοξενήσουν» εξυπηρετητές σε αυτήν την γραμμή. Επίσης οι ταυτόχρονες συνδέσεις που διαχειρίζεται κάθε εξυπηρετητής υπόκεινται σε περιορισμούς. Έτσι ανάλογα και με τις ανάγκες κάθε εξυπηρετητή έχουμε διακόσους χρήστες για τον διακομιστή κατάστασης και τον κεντρικό διακομιστή και είκοσι χρήστες για τον εξυπηρετητή παιχνιδιού.

Σε γενικές γραμμές η παρούσα απόδοση θα μπορούσε να θεωρηθεί ικανοποιητική καθώς τα όποια σφάλματα, προκύπτουν σπάνια. Τα περιθώρια βελτίωσης βεβαία είναι υπαρκτά, ενώ ο συγκεκριμένος τομέας θα πρέπει να βελτιωθεί για να γίνει πιο σοβαρή η χρήση του προγράμματος.

2.4.2 Ασφάλεια (Security)



Σχήμα 2.4-2: Μη-λειτουργική απαίτηση «Ασφάλεια»

Η εφαρμογή σε θέματα ασφάλειας αναπτύχτηκε σε επίπεδα που θεωρήθηκαν απαραίτητα, παρ' όλα αυτά η ασφάλεια είναι ένα μεγάλο και πολύπλοκο κομμάτι στην αρχιτεκτονική εξυπηρετητή - πελάτη. Η εφαρμογή επιδέχεται περαιτέρω βελτίωσης στο

θέμα της ασφάλεια στο μέλλον. Οι τεχνικές οι οποίες χρησιμοποιήθηκαν στο πρόγραμμα μας ήταν:

α) Πιστοποίηση ή επαλήθευση ταυτότητας (Authentication), για τον έλεγχο πρόσβασης στον κεντρικό εξυπηρετητή. Για να το πετύχουμε αυτό κατά την εκκίνηση της εφαρμογής και μετά την οθόνη υποδοχής εμφανίζεται ένα παράθυρο, στο οποίο ο χρήστης εισάγει ένα συνθηματικό και ένα κωδικό πρόσβασης τα οποία βρίσκονται αποθηκευμένα σε μια βάση η οποία βρίσκεται στο ίδιο λειτουργικό σύστημα με τον κεντρικό εξυπηρετητή. Εφόσον γίνει έλεγχος των στοιχείων που εισήγαγε ο χρήστης με αυτά της βάσης, αν τα δεδομένα είναι ταυτόσημα, ο χρήστης μεταβαίνει στο παράθυρο της κεντρικής οθόνης. Επίσης κάθε εξυπηρετητής παιχνιδιού που εκτελείται θα πρέπει πρώτα να πιστοποιηθεί από τον κεντρικό εξυπηρετητή για να διαπιστωθεί ότι έχει δικαιώματα για την υποστήριξη τις συγκεκριμένης λειτουργίας. Μετά την επιτυχή πιστοποίηση ο χρήστης μπορεί να κάνει τις απαραίτητες ρυθμίσεις για την εκκίνηση ενός τραπεζιού, που θα φιλοξενήσει τους ενδιαφερόμενους παίκτες. Η πιστοποίηση γίνεται καθαρά για λόγους αναγνώρισης και αντιστοίχησης της εφαρμογής με τον χρήστη από τον οποίο εκτελείται, προκείμενου να χρησιμοποιηθούν τα στοιχεία του για την εκτέλεση των κανόνων του παιχνιδιού σε επόμενη φάση.

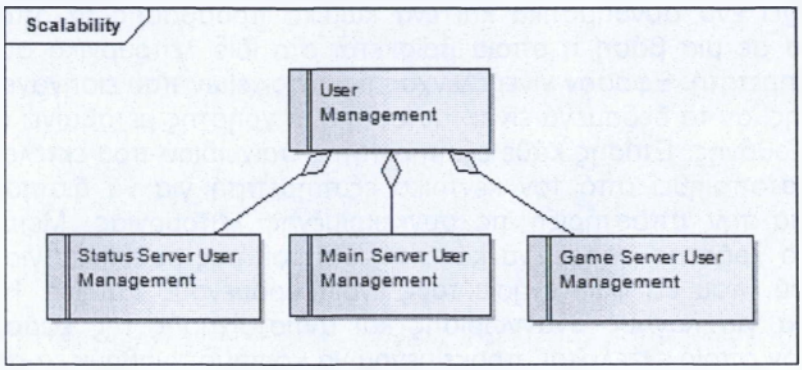
β) Εξουσιοδότηση (Authorization), για να επιτρέπουμε την πρόσβαση στα δεδομένα και τις λειτουργιές του προγράμματος, μόνο σε χρήστες στους οποίους επιτρέπεται να τις χρησιμοποιούν. Για την εφαρμογή της συγκεκριμένης λειτουργίας κατά την διαδικασία της ταυτοποίησης ο κεντρικός εξυπηρετητής δημιουργεί ένα αναγνωριστικό μιας χρήσης το οποίο και γνωστοποιεί στην εφαρμογή πελάτη(Client). Αυτό το αναγνωριστικό το γνωρίζει μόνο η εφαρμογή και όχι ο χρήστης. Ο κεντρικός εξυπηρετητής χρησιμοποίει αυτό το αναγνωριστικό για να «συστήσει» έναν χρήστη σε έναν εξυπηρετητή παιχνιδιού οπότε αυτό χρειαστεί. Έτσι διασφαλίζεται: (1) ότι δεν θα προσπαθήσει κάποιος χρήστης να οικειοποιηθεί την θέση ενός άλλου χρήστη και να ενεργεί για λογαριασμό του χωρίς ο ίδιος να το γνωρίζει και (2) ότι ο κωδικός χρήστη ο οποίος είναι ένα «ευαίσθητο» προσωπικό δεδομένο, δεν θα γνωστοποιείται στους εξυπηρετητές παιχνιδιού, και αυτό γιατί οι συγκεκριμένοι διακομιστές είναι χαμηλότερης εμπιστοσύνης σε σύγκριση με τον κεντρικό εξυπηρετητή.

Κατά την δημιουργία της εφαρμογής έγινε μια προσπάθεια υλοποίησης μιας τεχνικής αποτύπωσης των συναλλαγών από την επικοινωνία των εξυπηρετητών. Αυτή η απλοϊκή υλοποίηση της τεχνικής αποτύπωσης των συναλλαγών δεν προσδίδει σημαντικό αντίκτυπο στον τομέα της ασφάλειας του προγράμματος αυτήν την στιγμή, λόγω της μικρής χρήσης που γίνεται στην δεδομένη φάση. Όμως με καθολική εφαρμογή της τεχνικής σε όλες τις συναλλαγές των εξυπηρετητών είτε μεταξύ τους είτε με τους χρήστες, θα αποτελούσε μια ουσιαστική και χρήσιμη παθητική τεχνική ασφάλειας και ελέγχου λειτουργίας του συστήματος μας, δίνοντας πολύτιμα στοιχεία για την καλύτερη βελτίωση του προγράμματος.

Το εργαλείο το οποίο χρησιμοποιούμε (Lidgren network) για την επικοινωνία των πελατών με τους εξυπηρετητές μας παρέχει τη δυνατότητα για την ασφαλή κρυπτογράφηση των δεδομένων με την χρήση των πιο γνωστών αλγορίθμων κρυπτογράφησης. Λόγω της παραδοχής ότι το σύστημα μας δεν θα έχει κακόβουλους χρήστες, που θα επιχειρήσουν να το βλάψουν για προσωπικό τους όφελος, θεωρούμε

άσκοπο να ασχοληθούμε με το κομμάτι της κρυπτογράφησης δεδομένων στην παρούσα φάση.

2.4.3 Επεκτασιμότητα (Scalability)



Σχήμα 2.4-3: Μη-λειτουργική απαίτηση «Επεκτασιμότητα»

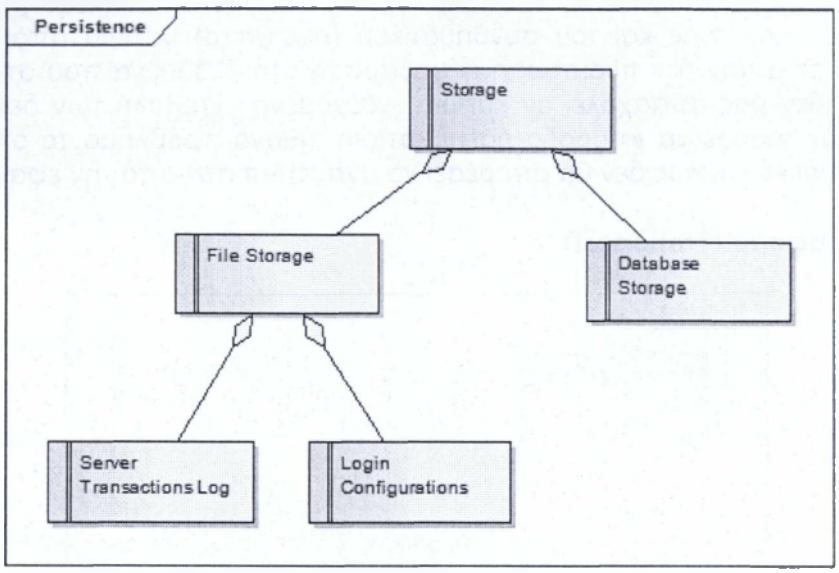
Οι διακομιστές που αναπτύχτηκαν στα πλαίσια της εκπόνησης της πτυχιακής εργασίας υπόκεινται σε περιορισμούς που αφορούν των αριθμό τον ταυτόχρονων συνδέσεων που μπορούν να εξυπηρετήσουν.

Ο εξυπηρετητής κατάστασης έχει ως μέγιστο όριο ταυτόχρονων συνδέσεων τους διακόσιους χρήστες. Ο αριθμός που τέθηκε για αυτόν τον διακομιστή είναι πολύ μεγαλύτερος από αυτόν που πραγματικά χρειάζεται η εφαρμογή, τουλάχιστον μέχρι το πέρας του σταδίου των δοκιμών.

Ο κεντρικός διακομιστής έχει και αυτός όριο ταυτόχρονων συνδέσεων τους διακόσιους χρήστες. Αν και αυτό το όριο είναι πιο εφικτό να καλυφτεί, λαμβάνοντας υπόψη ότι ο εξυπηρετητής θα «εκτελείται» πάνω σε προσωπικούς υπολογιστές καθώς και τις δυνατότητες αυτών, τέθηκε αυτό το όριο για την κάλυψη των δοκιμών. Θεωρείται απαραίτητο αυτό το όριο να μεγαλώσει σε περίπτωση που η εφαρμογή θα χρησιμοποιηθεί σε υλικό με καλυτέρα χαρακτηριστικά από αυτό των προσωπικών υπολογιστών σε μεταγενέστερο από το στάδιο ανάπτυξης και δοκίμων.

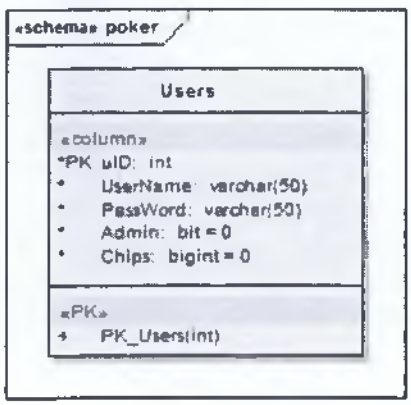
Για τον εξυπηρετητή παιχνιδιού ο μέγιστος αριθμός των ταυτόχρονων συνδέσεων είναι είκοσι χρήστες, δίνοντας έτσι τη δυνατότητα να «φιλοξενούνται» περισσότερες εφαρμογές του διακομιστή πάνω σε έναν εξυπηρετητή, ο οποίος σύμφωνα με τη λογική μας θα είναι προσωπικός υπολογιστής και σε πολλές περιπτώσεις δεν θα έχει ως μοναδικό στόχο την «φιλοξενία» της εφαρμογής.

2.4.4 Επιμονή (Persistence)



Σχήμα 2.4-4: Μη-λειτουργική απαίτηση «Επιμονή»

Για την μόνιμη αποθήκευση των δεδομένων που παράγονται κατά την διάρκεια εκτέλεσης της εφαρμογής μας καθώς και των δεδομένων που βρίσκονται προ-αποθηκευμένα και απαιτούνται για την σωστή και ουσιώδη «εκτέλεση» της εφαρμογής μας χρησιμοποιούμε δυο τεχνικές:



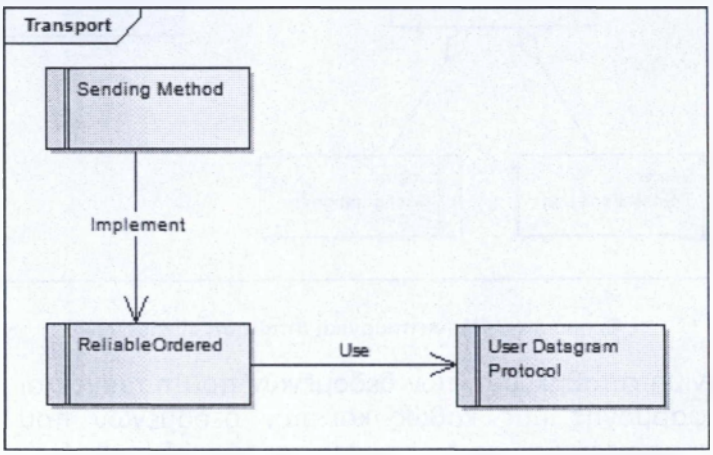
Σχήμα 2.4-5: Πίνακας Χρηστών

α) Βάση δεδομένων που περιέχει έναν πίνακα όπου βρίσκονται αποθηκευμένοι οι χρήστες που έχουν πρόσβαση στην εφαρμογή. Για κάθε χρήστη υπάρχουν τα απαραίτητα στοιχεία για την σύνδεση του, καθώς και πληροφορίες σχετικά με το βαθμό πρόσβασης στις λειτουργίες της εφαρμογής, το ποσό των εικονικών χρημάτων (μάρκες) που διαθέτει, το όνομα του και το εικονίδιο που θα χρησιμοποιηθεί από την εφαρμογή πελάτη για την αναγνώριση του από του άλλους χρήστες. Η βάση χρησιμοποιείται μόνο για ανάγνωση προ-αποθηκευμένων δεδομένων από τον κεντρικό διακομιστή. Στην παρούσα φάση της ανάπτυξης της εφαρμογής τα δεδομένα που βρίσκονται στην βάση δεδομένων δεν κρυπτογραφούνται καθ' ότι δεν θεωρούνται «ευαίσθητα» προσωπικά δεδομένα. Τα δεδομένα παράγονται από τους διαχειριστές και δεν έχουν κάποια συσχέτιση με τους χρήστες και η απώλεια τους δεν πρόκειται να τους επηρεάσει. Επίσης τα δεδομένα της δεν μεταβάλλονται από την «εκτέλεση» της εφαρμογής.

β) Ο διακομιστής κατάστασης χρησιμοποιεί αρχεία κειμένου με σκοπό την μόνιμη αποθήκευση πληροφοριών σχετικά με τις συνεδρίες μεταξύ αυτού και των χρηστών. Τα δεδομένα αποθηκεύονται σε αρχεία που είναι οργανωμένα σε φακέλους σύμφωνα με το

χρονικό διάστημα μέσα στο οποίο πραγματοποιήθηκαν οι συνεδρίες. Επίσης η εφαρμογή πελάτη χρησιμοποιεί την ίδια τεχνική για την αποθήκευση του αριθμού έκδοσης της εφαρμογής και του συνθηματικού (password) ύστερα από επιλογή του χρήστη. Και σε αυτήν την περίπτωση η πρόσβαση στα δεδομένα που αποθηκεύονται από τρίτους δεν μας απασχολεί αν και μια ενδεχόμενη μεταβολή των δεδομένων του αρχείου θα μπορούσε να «πυροδοτήσει» κάποιο πιθανό πρόβλημα το οποίο δεν έχει προβλεφθεί και επομένως δεν θα μπορέσει να αντιμετωπιστεί από την εφαρμογή.

2.4.5 Μεταφορά (Transport)



Σχήμα 2.4-6: Μη-λειτουργική απαίτηση «Μεταφορά»

Για την επίτευξη της διαδικασίας της μεταφοράς των δεδομένων στο σύστημα μας γίνεται χρήση των εργαλείων της βιβλιοθήκης `Lidgren network`. Η βιβλιοθήκη παρέχει μια σειρά μεθόδων για την μεταφορά των δεδομένων καθώς και εργαλεία για τον έλεγχο του προγράμματος κάτω από διαφορετικές συνθήκες. Η μέθοδος που χρησιμοποιήθηκε στην συγκεκριμένη περίπτωση για την υλοποίηση της επικοινωνίας μεταξύ των εξυπηρετητών και των πελατών είναι η `ReliableOrdered`. Με αυτή τη μέθοδο εξασφαλίζουμε ότι το μήνυμα θα φτάσει στον προορισμό του χωρίς να υπάρχει απώλεια πακέτων και ότι τα πακέτα θα φτάσουν με την σειρά με την οποία στάλθηκαν. Η μέθοδος αυτή αποτελεί τη βέλτιστη λύση σύμφωνα με τις απαιτήσεις της εφαρμογής μας.

-Κεφάλαιο 3ο-

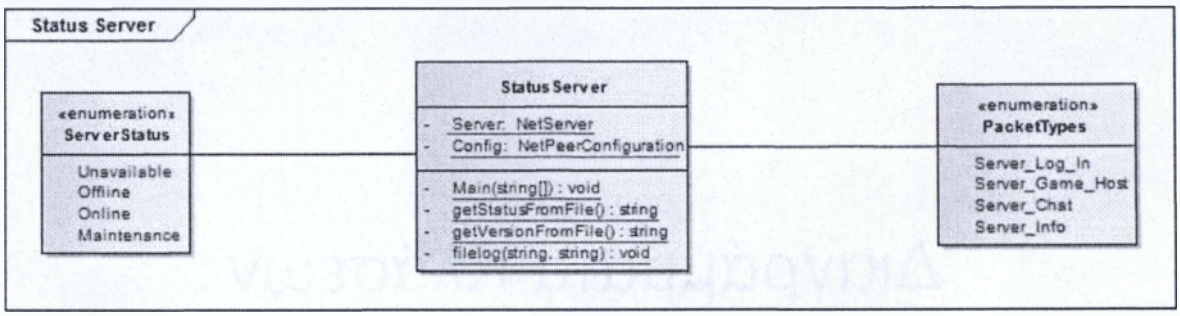
Διαγράμματα Κλάσεων

3. Διαγράμματα Κλάσεων

Σε αυτό το κεφάλαιο θα γίνει απεικόνιση των κλάσεων που υλοποιήθηκαν στα πλαίσια της εργασίας για την δημιουργία των προγραμμάτων, καθώς και των συσχετίσεων μεταξύ τους. Η γραφική απεικόνιση έγινε με την χρήση των εργαλείων της UML[10] και των κανόνων που αυτή ορίζει. Συνοπτικά έχουμε αναπαράσταση των κλάσεων μέσω γεωμετρικών σχημάτων. Ενώ για την αναπαράσταση των συσχετίσεων χρησιμοποιούνται διαφόρων ειδών γραμμές.

3.1 Εξυπηρετητής Κατάστασης

Ο εξυπηρετητής κατάστασης στην παρούσα φάση έχει λιτή υλοποίηση παρ' όλα αυτά διαθέτει προοπτικές για περαιτέρω ανάπτυξη και εξέλιξη. Την δεδομένη χρονική στιγμή ο εξυπηρετητής κατάστασης αποτελείται από τα εξής στοιχεία:



Σχήμα 3.1-1: Διάγραμμα κλάσεων Εξυπηρετητή Κατάστασης

`ServerStatus`
κλάση απαρίθμησης, χρησιμοποιείται για την απαρίθμηση των διαφορετικών καταστάσεων των εξυπηρετητών.

`PacketTypes`
κλάση απαρίθμησης, σκοπός της είναι η «περιτύλιξη» των μηνυμάτων που λαμβάνονται και αποστέλλονται από τον εξυπηρετητή.

`StatusServer`
κεντρική κλάση που περιέχει τα παρακάτω πεδία και μεθόδους:

`NetServer Server`: πεδίο, είναι βασική κλάση της βιβλιοθήκης «Lidgren network» και περιέχει όλες τις πληροφορίες και τις λειτουργίες που αφορούν στον εξυπηρετητή.

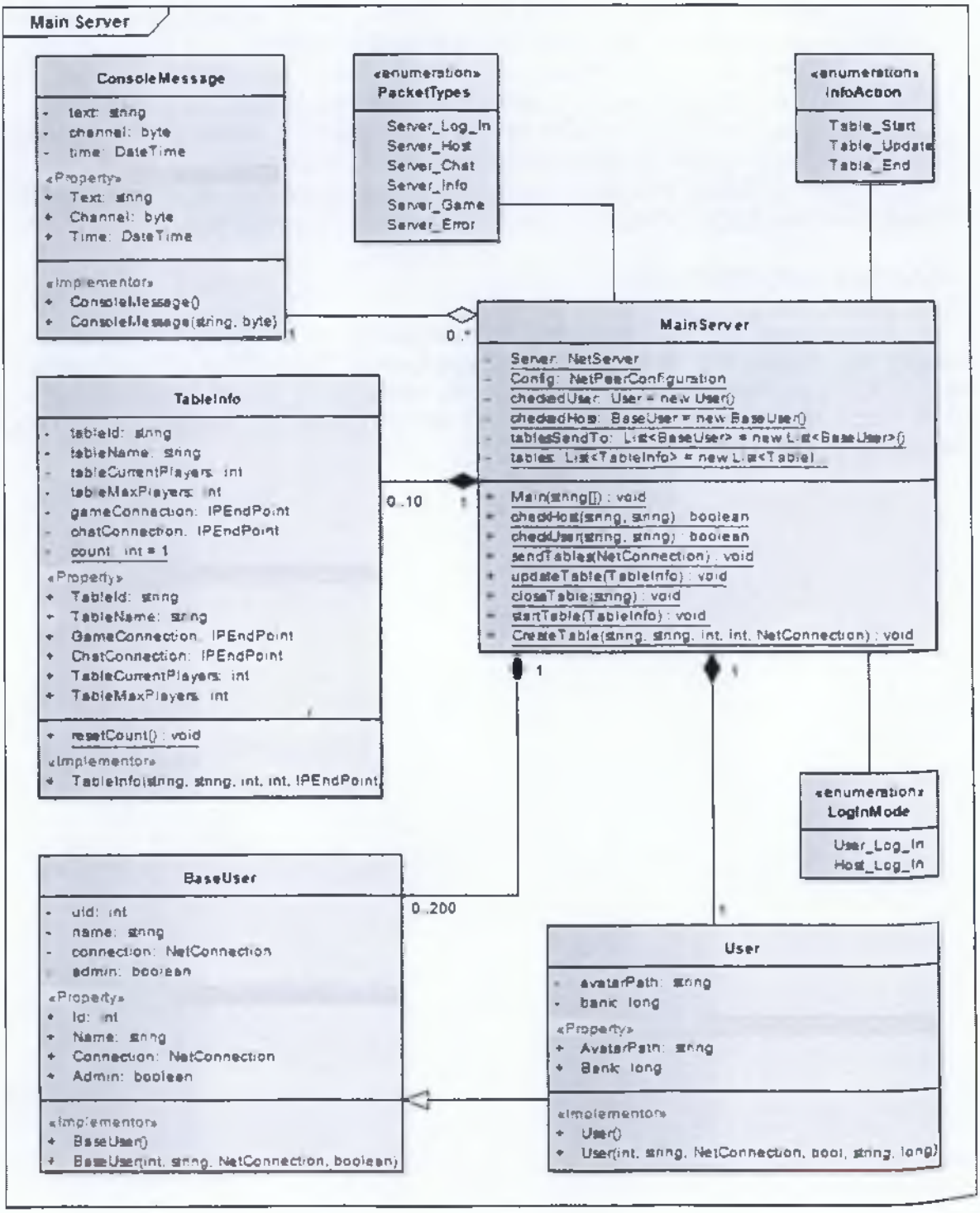
`NetPeerConfiguration Config`: πεδίο, σε αυτό το αντικείμενο αποθηκεύονται πληροφορίες που αφορούν στην επικοινωνία του εξυπηρετητή.

[10] Unified Modeling Language (Ενοποιημένη Γλώσσα Μοντελοποίησης): είναι η πρότυπη γλώσσα μοντελοποίησης στην τεχνολογία λογισμικού. Χρησιμοποιείται για τη γραφική απεικόνιση, τον προσδιορισμό, την κατασκευή και την τεκμηρίωση των στοιχείων ενός συστήματος λογισμικού. Μπορεί να χρησιμοποιηθεί σε διάφορες φάσεις ανάπτυξης, από την ανάλυση απαιτήσεων ως τον έλεγχο ενός ολοκληρωμένου συστήματος.

`Main`: μέθοδος, η κεντρική μέθοδος από την όποια γίνεται η κλήση των υπολοίπων μεθόδων και η διαχείριση του προγράμματος του εξυπηρετητή.
`getStatusFromFile`: μέθοδος, η μέθοδος αυτή χρησιμοποιείται για την ανάγνωση πληροφοριών από το αρχείο ρυθμίσεων που διαθέτει ο εξυπηρετητής.
`getVersionFromFile`: μέθοδος με την οποία γίνεται η ανάγνωση της έκδοσης του προγράμματος από το αρχείο ρυθμίσεων του εξυπηρετητή.
`filelog`: μέθοδος που είναι υπεύθυνη για την καταγραφή σε αρχείο κειμένου όλων των συνεδριών μεταξύ του εξυπηρετητή και των πελατών του.

3.2 Κεντρικός Εξυπηρετητής

Ο κεντρικός εξυπηρετητής είναι το πρόγραμμα το οποίο έχει επωμιστεί την ευθύνη της διαχείρισης των χρηστών. Συγκεκριμένα διαχειρίζεται την επικοινωνία μεταξύ των χρηστών και τις πληροφορίες που αφορούν τα ενεργά τραπέζια. Για την υλοποίηση των παραπάνω διαδικασιών αναπτύχτηκαν οι παρακάτω κλάσεις. Ακολουθεί σχήμα στην επόμενη σελίδα.



Σχήμα 3.2-1: Διάγραμμα κλάσεων Κεντρικού Εξυπηρετητή

PacketTypes
κλάση απαρίθμησης, σκοπός της είναι η «περιτύλιξη» των μηνυμάτων που λαμβάνονται και αποστέλλονται από τον εξυπηρετητή.

LogInMode
κλάση απαρίθμησης, χρησιμοποιείται για τον διαχωρισμό των χρηστών που συνδέονται στον εξυπηρετητή.

InfoAction
κλάση απαρίθμησης, με αυτήν την κλάση απαρίθμησης «μαρκάρονται» τα μηνύματα που αφορούν στην δημιουργία, στην ανανέωση, και στον τερματισμό των ενεργών τραπεζιών.

TableAction
κλάση απαρίθμησης, μέσω αυτής της κλάσης γίνεται η αναγνώριση των μηνυμάτων που περιέχουν πληροφορίες σχετικά με την συμμετοχή ενός παίκτη ή την αποχώρηση του από κάποιο τραπέζι.

ConsoleMessage
κλάση, περιέχει πληροφορίες για μηνύματα που αφορούν στην επικοινωνία των χρηστών. Περιέχει τα εξής πεδία και μεθόδους:

`string text`: πεδίο, το μήνυμα το οποίο πληκτρολογήθηκε από τον χρήστη.
`byte channel`: πεδίο, το κανάλι επικοινωνίας στο οποίο αποστέλλεται το μήνυμα.
`DateTime time`: πεδίο, η χρονική στιγμή κατά την οποία συντάχτηκε το μήνυμα.
`ConsoleMessage`: μέθοδος, δυο μέθοδοι που φέρουν το όνομα της κλάσης και χρησιμοποιούνται για την δημιουργία αντικειμένων αυτής.

TableInfo
κλάση, περιλαμβάνει πληροφορίες σχετικά με τα ενεργά τραπέζια του συστήματος. Αποτελείται από τα παρακάτω πεδία και μεθόδους:

`string tableId`: πεδίο, το μοναδικό αναγνωριστικό κάθε τραπεζιού.
`string tableName`: πεδίο, η ονομασία του τραπεζιού.
`IpEndPoint gameConnection`: πεδίο, τα στοιχεία για την σύνδεση στο εξυπηρετητή παιχνιδιού που φιλοξενεί το συγκεκριμένο τραπέζι.
`IpEndPoint chatConnection`: πεδίο, τα στοιχεία για την επικοινωνία με τον κεντρικό εξυπηρετητή.
`int tableCurrentPlayers`: πεδίο, ο αριθμός των παικτών που βρίσκονται στο τραπέζι.
`int tableMaxPlayers`: πεδίο, ο μέγιστος αριθμός των παικτών που φιλοξενεί το συγκεκριμένο τραπέζι.
`int count`: πεδίο, χρησιμοποιείται για την δημιουργία του μοναδικού αναγνωριστικού των τραπεζιών.
`resetCount`: μέθοδος, επαναφέρει τον μετρητή για την δημιουργία μοναδικού αναγνωριστικού των τραπεζιών.

`TableInfo`: μέθοδος, χρησιμοποιείται για την δημιουργία αντικειμένων της συγκεκριμένης κλάσης.

`BaseUser`

κλάση, απεικονίζει τις πληροφορίες που αφορούν τους χρήστες της εφαρμογής σε ένα αφαιρετικό επίπεδο. Διαθέτει τα παρακάτω πεδία και τις μεθόδους.

`int uId`: πεδίο, το μοναδικό αναγνωριστικό κάθε αντικειμένου της κλάσης.

`string name`: πεδίο, το όνομα του χρήστη.

`NetConnection connection`: πεδίο, τα στοιχεία σύνδεσης στην εφαρμογή πελάτη του χρήστη.

`boolean admin`: πεδίο, τα δικαιώματα πρόσβασης του χρήστη στις λειτουργίες της εφαρμογής.

`BaseUser`: μέθοδος, δυο μέθοδοι που φέρουν το ίδιο όνομα με την κλάση και χρησιμοποιούνται για την δημιουργία αντικειμένων.

`User`

κλάση, κληρονομεί από την κλάση BaseUser και την επεκτείνει προσθέτοντας πεδία για τις πληροφορίες που απαιτούνται μετά την σύνδεση του χρήστη στο σύστημα.

`string avatarPath`: πεδίο, η διαδρομή που αντιστοιχεί στην εικόνα προσωπείου του χρήστη μέσα στο σύστημα.

`long bank`: πεδίο, οι συνολικές μάρκες που διαθέτει ο χρήστης στην κατοχή του.

`User`: μέθοδος, δυο μέθοδοι που έχουν το ίδιο όνομα με την κλάση και χρησιμοποιούνται για την δημιουργία αντικειμένων.

`MainServer`

κλάση, η κλάση του κεντρικού εξυπηρετητή, η οποία εκτελεί τις διεργασίες και τις λειτουργίες αυτού. Αποτελείται από τα παρακάτω πεδία και μεθόδους:

`NetServer Server`: πεδίο, είναι η βασική κλάση της βιβλιοθήκης «Lidgren network» και περιέχει όλες τις πληροφορίες και τις λειτουργίες που αφορούν στον εξυπηρετητή.

`NetPeerConfiguration Config`: σε αυτό το αντικείμενο αποθηκεύονται πληροφορίες που αφορούν στην επικοινωνία του εξυπηρετητή.

`User checkedUser`: πεδίο, σε αυτή τη μεταβλητή αποθηκεύονται τα στοιχεία του χρήστη μετά την επιτυχημένη σύνδεση του στο σύστημα, για μελλοντική χρήση τους.

`BaseUser checkedHost`: πεδίο, σε αυτή τη μεταβλητή αποθηκεύονται τα στοιχεία του χρήστη με δικαίωμα φιλοξενίας παιχνιδιών μετά την επιτυχημένη σύνδεση του στο σύστημα, για μελλοντική χρήση τους.

`List<BaseUser> tablesSendTo`: πεδίο, είναι η λίστα στην οποία εγγράφονται οι χρήστες για την συνεχή ενημέρωση τους με πληροφορίες σχετικά με την έναρξη, μεταβολή και τον τερματισμό των τραπεζιών.

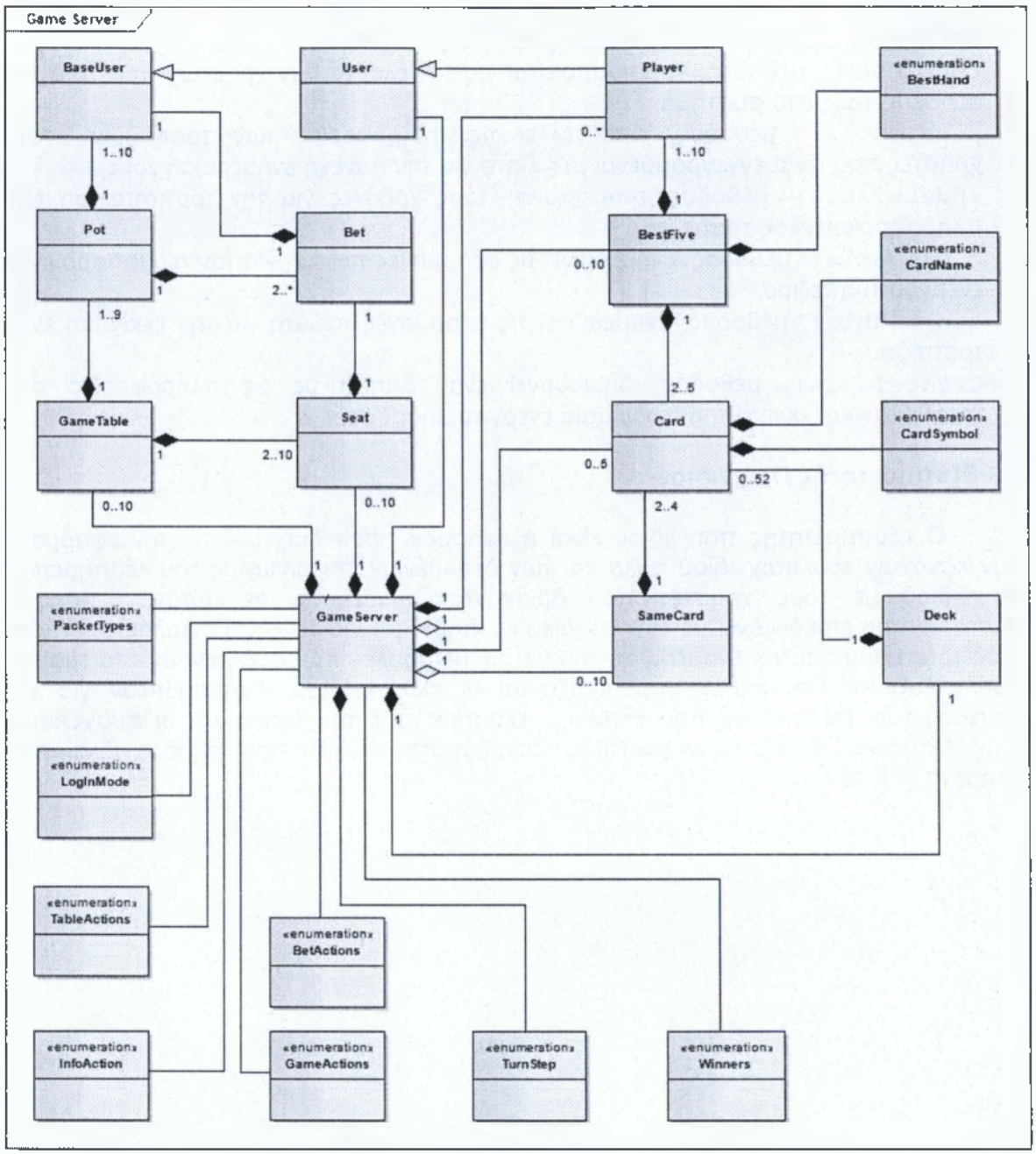
`List<Tableinfo> tables`: πεδίο, λίστα όπου αποθηκεύονται οι πληροφορίες των ενεργών τραπεζιών.

`Main`: μέθοδος, η κεντρική μέθοδος από την οποία γίνεται η κλήση των υπολοίπων μεθόδων και η διαχείριση του προγράμματος του εξυπηρετητή.

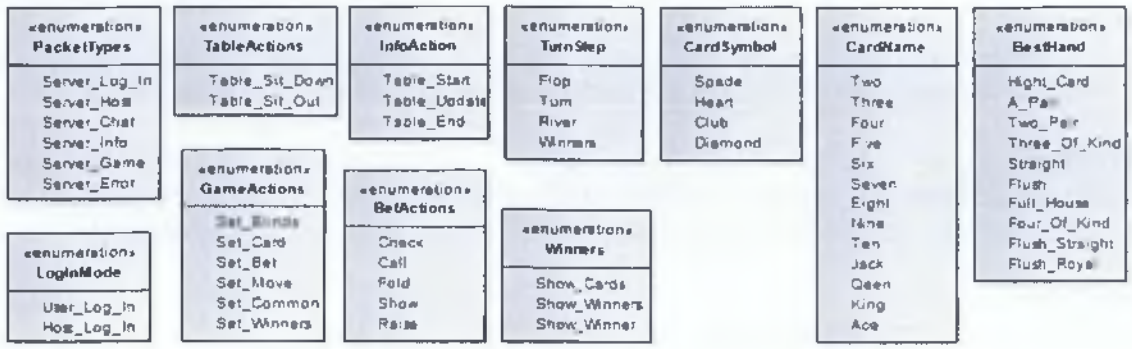
`checkHost`: μέθοδος, χρησιμοποιείται για τον έλεγχο πριν από την σύνδεση στο σύστημα των χρηστών με δικαίωμα φιλοξενίας παιχνιδιών.
`checkUser`: μέθοδος, χρησιμοποιείται για τον έλεγχο των χρηστών πριν από την σύνδεση τους στο σύστημα.
`sendTables`: μέθοδος, αποστέλλει τις πληροφορίες των τραπεζιών στους χρήστες που είναι εγγεγραμμένοι στη λίστα για την συνεχή ενημέρωση τους.
`updateTable`: μέθοδος, ενημερώνει τους χρήστες για την τροποποίηση των πληροφοριών ενός τραπεζιού.
`closeTable`: μέθοδος, ενημερώνει τις εφαρμογές πελάτη για τον τερματισμό ενός ενεργού τραπεζιού.
`startTable`: μέθοδος, ενημερώνει τις εφαρμογές πελάτη για την εκκίνηση ενός τραπεζιού.
`CreateTable`: μέθοδος, δημιουργεί ένα τραπέζι με τις πληροφορίες που ζητήθηκαν και εκκινεί την διαδικασία ενεργοποίησης του.

3.3 Εξυπηρετητής Παιχνιδιού

Ο εξυπηρετητής παιχνιδιού είναι η εφαρμογή που διαχειρίζεται την εφαρμογή των κανόνων του παιχνιδιού αλλά και των δεδομένων επικοινωνίας του εξυπηρετητή παιχνιδιού με τους χρήστες που βρίσκονται συνδεμένοι σε αυτόν. Επίσης ο εξυπηρετητής επικοινωνεί με τον κεντρικό εξυπηρετητή για τη λήψη εντολής εκκίνησης ενός τραπεζιού και την ενημέρωση του για τις μεταβολές που συμβαίνουν στο τραπέζι που φιλοξενεί. Παρακάτω παρουσιάζονται οι κλάσεις που υλοποιήθηκαν για την επίτευξη των διεργασιών που εκτελεί ο εξυπηρετητής παιχνιδιού και οι συσχετισμοί τους. Ακολουθεί σχήμα του συνοπτικού διαγράμματος του εξυπηρετητή παιχνιδιού στην επόμενη σελίδα.



Σχήμα 3.3-1: Συνοπτικό διάγραμμα κλάσεων Εξυπηρετητή Παιχνιδιού



Σχήμα 3.3-2: Κλάσεις Απαρίθμησης του Εξυπηρετητή Παιχνιδιού

PacketTypes
κλάση απαρίθμησης, σκοπός της είναι η «περιτύλιξη» των μηνυμάτων που λαμβάνονται και αποστέλλονται από τον εξυπηρετητή.

LogInMode
κλάση απαρίθμησης, χρησιμοποιείται για τον διαχωρισμό των χρηστών που συνδέονται στον εξυπηρετητή.

InfoAction
κλάση απαρίθμησης, με αυτήν την κλάση απαρίθμησης «μαρκάρονται» τα μηνύματα που αφορούν την δημιουργία, ανανέωση, και τον τερματισμό των ενεργών τραπεζιών.

TableAction
κλάση απαρίθμησης, μέσω αυτής της κλάσης γίνεται η αναγνώριση των μηνυμάτων που περιέχουν πληροφορίες σχετικά με την συμμετοχή ενός παίκτη ή την αποχώρηση του από κάποιο τραπέζι.

GameActions
κλάση απαρίθμησης, με χρήση της παρούσας κλάσης διαχωρίζονται τα «πακέτα» που περιέχουν δεδομένα σχετικά με τις φάσεις (τυφλό ποντάρισμα, μοίρασμα φύλλων, ποντάρισμα, κ.τ.λ) του παιχνιδιού.

BetActions
κλάση απαρίθμησης, απαριθμεί όλες τις δυνατές επιλογές και κινήσεις κάθε παίκτη με τα φύλλα που έχει στην κατοχή του.

TurnStep
κλάση απαρίθμησης, χρησιμοποιείται για την αρίθμηση των φάσεων της παρτίδας από το άνοιγμα των κοινών φύλλων μέχρι την ανακοίνωση του νικητή.

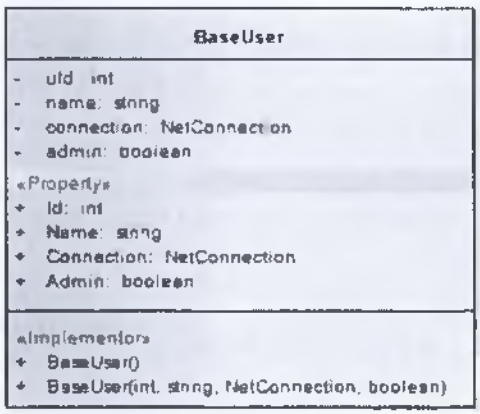
Winners
κλάση απαρίθμησης, με αυτήν την κλάση γίνεται η αναγνώριση των μηνυμάτων που περιέχουν πληροφορίες σχετικά με τις διαδικασίες ανακοίνωσης των νικητών.

BestHand
κλάση απαρίθμησης, περιέχει τις διακριτές ονομασίες των πιθανών σχημάτων που μπορούν να δημιουργηθούν από έναν παίκτη. Περιέχεται στην κλάση BestHand.

CardName
κλάση απαρίθμησης, οι ονομασίες των καρτών της τράπουλας.

CardSymbol
κλάση απαρίθμησης, τα τέσσερα σύμβολα των καρτών της τράπουλας.

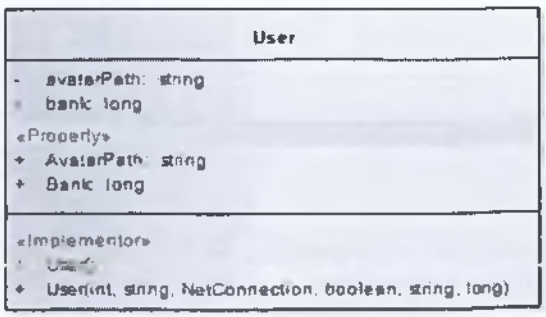
BaseUser
κλάση, απεικονίζει τις πληροφορίες που αφορούν στους χρήστες της εφαρμογής σε αφαιρετικό επίπεδο. Διαθέτει τα παρακάτω πεδία και μεθόδους.



Σχήμα 3.3-3: Κλάση BaseUser

`int uId`: πεδίο, το μοναδικό αναγνωριστικό κάθε αντικειμένου της κλάσης.
`string name`: πεδίο, το όνομα του χρήστη.
`NetConnection connection`: πεδίο, τα στοιχεία σύνδεσης στην εφαρμογή πελάτη του χρήστη.
`boolean admin`: πεδίο, τα δικαιώματα πρόσβασης του χρήστη στις λειτουργίες της εφαρμογής.
`BaseUser`: μέθοδος, δυο μέθοδοι που φέρουν το ίδιο όνομα με την κλάση και χρησιμοποιούνται για την δημιουργία και αρχικοποίηση των αντικειμένων αυτής.

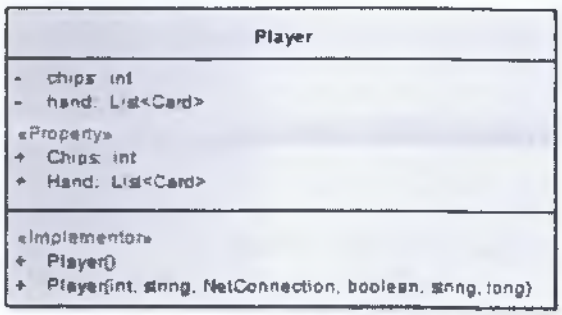
User



Σχήμα 3.3-4: Κλάση User

κλάση, κληρονομεί από την κλάση BaseUser και την επεκτείνει προσθέτοντας πεδία για τις πληροφορίες που απαιτούνται μετά την σύνδεση του χρήστη στο σύστημα.
`string avatarPath`: πεδίο, η διαδρομή που αντιστοιχεί στην εικόνα προσωπείου του χρήστη μέσα στο σύστημα.
`long bank`: πεδίο, οι συνολικές μάρκες που διαθέτει ο χρήστης στην κατοχή του.
`User`: μέθοδος, δυο μέθοδοι που έχουν το ίδιο όνομα με την κλάση και χρησιμοποιούνται για την δημιουργία και αρχικοποίηση των αντικειμένων αυτής.

Player



Σχήμα 3.3-5: Κλάση Player

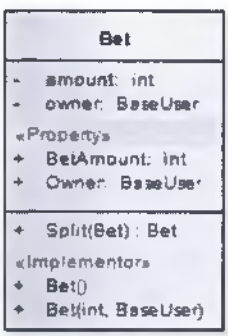
κλάση, κληρονομεί από την κλάση User και την επεκτείνει προσθέτοντας πεδία για τις πληροφορίες που απαιτούνται προκειμένου να μπορεί ένας χρήστης να συμμετέχει στα τραπέζια.

`long chips`: πεδίο, οι μάρκες που κατέχει ο χρήστης κατά την δεδομένη χρονική στιγμή στο συγκεκριμένο τραπέζι της εφαρμογής.

`List<Card> hand`: πεδίο, περιέχει μια λίστα με τις πληροφορίες για τις κάρτες που βρίσκονται στην κατοχή του παίκτη.

`Player`: μέθοδος, δυο μέθοδοι με όνομα ίδιο με αυτό της κλάσης που χρησιμοποιούνται για την δημιουργία και αρχικοποίηση των αντικειμένων αυτής.

Bet



Σχήμα 3.3-6: Κλάση Bet

κλάση, η κλάση αυτή διατηρεί πληροφορίες για τα πονταρίσματα που πραγματοποιούνται καθώς και τα στοιχεία των παικτών που τα πραγματοποιούν.

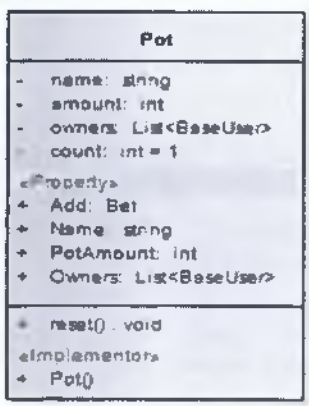
`int amount`: πεδίο, η αριθμητική τιμή του συγκεκριμένου πονταρίσματος.

`BaseUser owner`: πεδίο, οι πληροφορίες του παίκτη στον οποίο ανήκει το ποντάρισμα.

`Split`: μέθοδος, η μέθοδος αυτή χρησιμοποιείται για την διαίρεση του πονταρίσματος, στην περίπτωση που πρέπει να δημιουργηθεί μια πλαϊνή στοίβα πονταρίσματος (Side Pot).

`Bet`: μέθοδος, δυο μέθοδοι με όνομα ίδιο με αυτό της κλάσης που χρησιμοποιούνται για την δημιουργία και αρχικοποίηση των αντικειμένων αυτής.

Pot



Σχήμα 3.3-7: Κλάση Pot

κλάση, χρησιμοποιείται για την διατήρηση των πληροφοριών σχετικά με τα πονταρίσματα που γίνονται σε κάθε παρτίδα. Σε ένα τραπέζι μπορούν να υπάρχουν πολλά διαφορετικά στιγμιότυπα της κλάσης.

`string name`: πεδίο, το όνομα-αναγνωριστικό κάθε στοίβας πονταρίσματος που δημιουργείται.

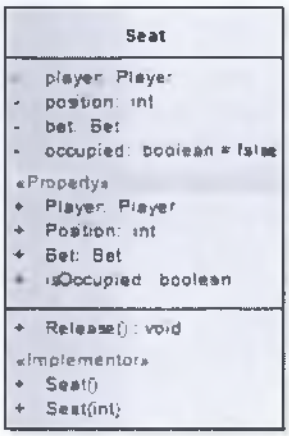
`int amount`: πεδίο, το άθροισμα των μαρκών που περιέχονται στη συγκεκριμένη στοίβα πονταρίσματος.

`List<BaseUser> owners`: πεδίο, όλοι οι χρήστες που έχουν συνεισφέρει στη στοίβα πονταρίσματος, και επομένως έχουν δικαίωμα απόκτησης της.

`int count`: πεδίο, απαριθμητής που χρησιμοποιείται για τη

δημιουργία μοναδικών ονομάτων για τα στιγμιότυπα της κλάσης.
`reset`: μέθοδος, χρησιμοποιείται για τον μηδενισμό του απαριθμητή count στο τέλος κάθε παρτίδας.
`Pot`: μέθοδος, μέθοδος για την δημιουργία αντικειμένων της συγκεκριμένης κλάσης.

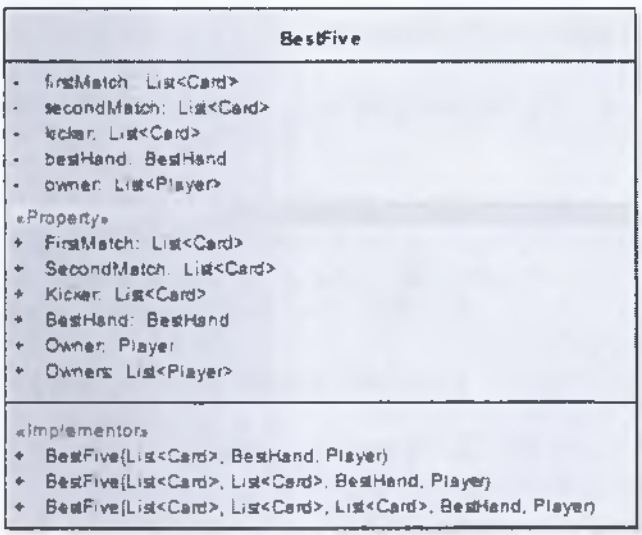
`Seat`
κλάση, συμβολίζει την θέση που κατέχει ο παίκτης όπως αυτή παρουσιάζεται στα τραπέζια παιχνιδιών.



Σχήμα 3.3-8: Κλάση Seat

`Player player`: πεδίο, τα στοιχεία του παίκτη ο οποίος κατέχει την θέση την δεδομένη χρονική στιγμή.
`int position`: πεδίο, ο αριθμός της θέσης στο τραπέζι όπως αυτή ορίζεται από το σύστημα μας.
`Bet bet`: πεδίο, σε αυτήν την μεταβλητή τοποθετούνται τα πονταρίσματα των παικτών μέχρι να μπουν στην στοίβα πονταρίσματος.
`bool occupied`: πεδίο, δυαδική μεταβλητή αλήθειας μέσω της οποίας ενημερωνόμαστε για το αν είναι κατειλημμένη ή ελεύθερη η συγκεκριμένη θέση.
`Release`: μέθοδος, «απελευθερώνει» τη συγκεκριμένη θέση στην περίπτωση που είναι κατειλημμένη.
`Seat`: μέθοδος, δυο μέθοδοι με όνομα ίδιο με αυτό της κλάσης που χρησιμοποιούνται για την δημιουργία και αρχικοποίηση των αντικειμένων αυτής.

`BestFive`

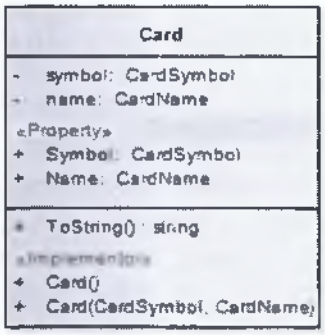


Σχήμα 3.3-9: Κλάση BestFive

κλάση, βαθμονομεί το φύλλο του παίκτη με τις κάρτες που διαθέτει στην κατοχή του και σε συνδυασμό με τα πέντε κοινά φύλλα που εμφανίζονται στο τραπέζι, βρίσκει την καλύτερη πεντάδα από αυτά.
`List<Card> firstMatch`: πεδίο, η πρώτη από τις τρεις λίστες στις οποίες μοιράζονται οι κάρτες ανάλογα με την διαβαθμισμένη αξία τους.
`List<Card> secondMatch`: πεδίο, η δεύτερη από τις τρεις λίστες στις οποίες μοιράζονται οι κάρτες ανάλογα με την διαβαθμισμένη αξία τους.
`List<Card> kicker`: πεδίο, η τρίτη από τις τρεις λίστες στις οποίες μοιράζονται οι κάρτες ανάλογα με την διαβαθμισμένη αξία τους.

`BestHand bestHand`: πεδίο, η βαθμονόμηση του φύλλου συμφώνα με τις διακριτές τιμές όπως αυτές ορίζονται στην κλάση απαρίθμησης BestHand.
`List<Player> owner`: πεδίο, οι πληροφορίες όλων των παικτών που διαθέτουν τη συγκεκριμένη καλύτερη πεντάδα.
`BestFive`: μέθοδος, τρεις μέθοδοι με όνομα ίδιο με αυτό της κλάσης που χρησιμοποιούνται για την δημιουργία και αρχικοποίηση των αντικειμένων αυτής.

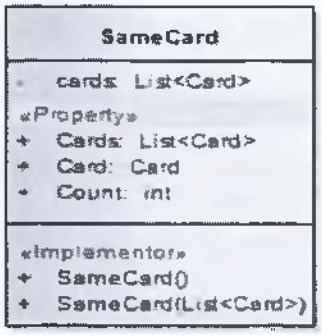
Card



Σχήμα 3.3-10: Κλάση Card

κλάση, αντιπροσωπεύει στο πρόγραμμα μας τις κάρτες μιας τράπουλας όπως αυτές παρουσιάζονται στον πραγματικό κόσμο.
`CardSymbol symbol`: πεδίο, ένα από τα τέσσερα πιθανά σύμβολα που μπορεί να έχει μια κάρτα.
`CardName name`: πεδίο, το όνομα της κάρτας σύμφωνα με την ονομασία που έχουν στον πραγματικό κόσμο.
`ToString`: μέθοδος, επιστρέφει την ονομασία και το σύμβολο της κάρτας.
`Card`: μέθοδος, δυο μέθοδοι με όνομα ίδιο με αυτό της κλάσης και σκοπό την δημιουργία και αρχικοποίηση αντικειμένων αυτής.

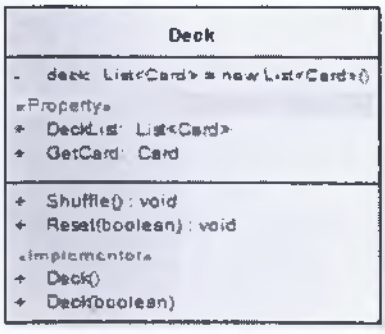
SameCard



Σχήμα 3.3-11: Κλάση SameCard

κλάση, μέσω αυτής της κλάσης γίνεται η οργάνωση των καρτών με την ίδια ονομασία κατά την διαδικασία αναζήτησης της καλύτερης πεντάδας για κάθε παίκτη.
`List<Card> cards`: πεδίο, η λίστα με τις κάρτες που έχουν την ίδια ονομασία.
`int Count`: πεδίο, ο αριθμός των καρτών με το ίδιο όνομα.
`SameCard`: μέθοδος, δυο μέθοδοι με ίδιο όνομα με αυτό της κλάσης που χρησιμοποιούνται για την δημιουργία και αρχικοποίηση αντικειμένων αυτής.

Deck



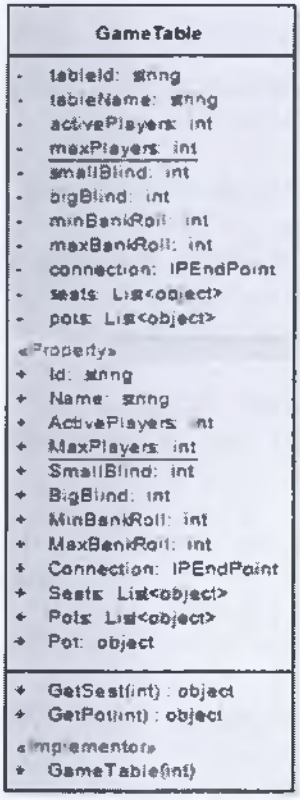
Σχήμα 3.3-12: Κλάση Deck

κλάση, περιέχει μια λίστα από κάρτες οι οποίες μαζί σχηματίζουν μια τράπουλα όπως αυτή ορίζεται στον πραγματικό κόσμο.
`List<Card> deck`: πεδίο, η λίστα στην οποία περιέχονται οι πληροφορίες των καρτών.
`Shuffle`: μέθοδος, ανακατεύει τις κάρτες της τράπουλας μέσω γεννήτριας τυχαίων αριθμών.
`Reset`: μέθοδος, επαναφέρει την τράπουλα είτε στην προεπιλεγμένη κατάσταση είτε με τυχαία σειρά.
`Deck`: μέθοδος, δυο μέθοδοι με ίδιο όνομα με αυτό της

κλάσης που χρησιμοποιούνται για την δημιουργία και αρχικοποίηση των αντικειμένων αυτής.

GameTable

κλάση, είναι η αναπαράσταση του τραπεζιού στο οποίο παίζεται το παιχνίδι.



Σχήμα 3.3-13: Κλάση GameTable

`string tableId`: πεδίο, το μοναδικό αναγνωριστικό του κάθε στιγμιότυπου της κλάσης.

`string tableName`: πεδίο, το όνομα της στιγμιότυπου της κλάσης.

`int activePlayers`: πεδίο, οι «ενεργοί» παίκτες στο συγκεκριμένο τραπέζι.

`int maxPlayers`: πεδίο, ο μέγιστος αριθμός των παικτών που μπορούν να συμμετάσχουν στο συγκεκριμένο τραπέζι.

`int smallBlind`: πεδίο, το μικρό τυφλό ποντάρισμα όπως αυτό είναι ορισμένο από τους κανόνες του παιχνιδιού.

`int bigBlind`: πεδίο, το μεγάλο τυφλό ποντάρισμα όπως αυτό είναι ορισμένο από τους κανόνες του παιχνιδιού.

`int minBankRoll`: πεδίο, το ελάχιστο ποσό που απαιτείται για την είσοδο του παίκτη στο τραπέζι.

`int maxBankRoll`: πεδίο, το μέγιστο ποσό με το οποίο μπορεί να εισέλθει ο παίκτης στο τραπέζι.

`IpEndPoint connection`: πεδίο, τα στοιχεία σύνδεσης στο συγκεκριμένο τραπέζι.

`List<object> seats`: πεδίο, η λίστα με τις θέσεις του τραπεζιού.

`List<object> pots`: πεδίο, η λίστα με τις στοίβες πονταρισμάτων του τραπεζιού.

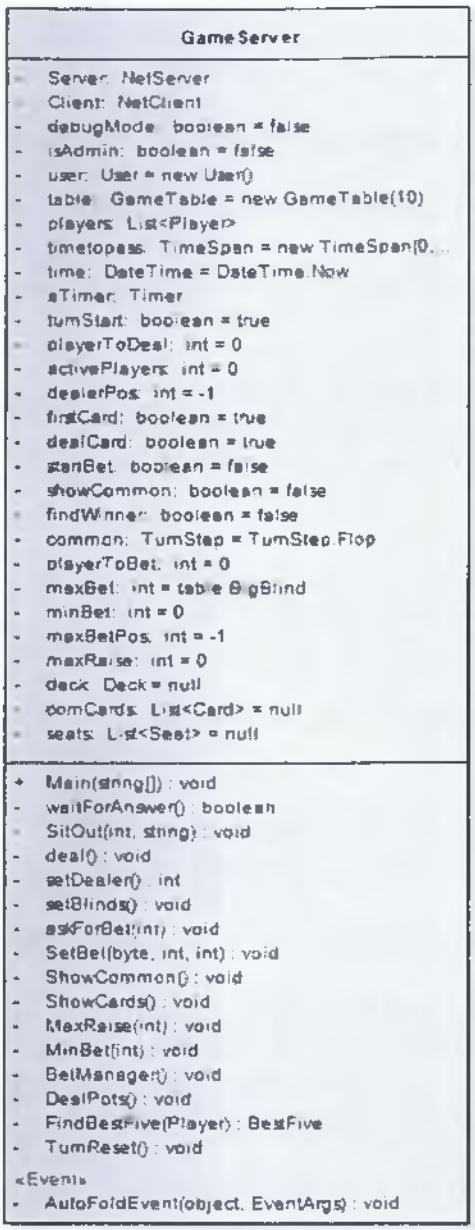
`GetSeat`: μέθοδος, επιστρέφει την αιτούμενη βάση αριθμού θέση.

`GetPot`: μέθοδος, επιστρέφει την αιτούμενη βάση αριθμού στοίβας πονταρίσματος.

`GameTable`: μέθοδος, μέθοδος για την δημιουργία και αρχικοποίηση αντικειμένων της κλάσης.

GameServer

κλάση, η κεντρική κλάση του εξυπηρετητή μέσω της οποίας καλούνται οι υπόλοιπες μέθοδοι της. Σκοπός της είναι η εκτέλεση των διαδικασιών του εξυπηρετητή.



Σχήμα 3.3-14: Κλάση GameServer

`NetServer Server`: πεδίο, είναι βασική κλάση της βιβλιοθήκης Lidgren network και περιέχει όλες τις πληροφορίες και τις λειτουργίες που αφορούν στον εξυπηρετητή.

NetClient Client: πεδίο, είναι βασική κλάση της βιβλιοθήκης Lidgren network και περιέχει όλες τις πληροφορίες και τις λειτουργίες που αφορούν στον πελάτη.

`bool debugMode`: πεδίο, δυαδική μεταβλητή αλήθειας που θέτει την εφαρμογή σε κατάσταση απασφαλμάτωσης.

`bool isAdmin`: πεδίο, δυαδική μεταβλητή αλήθειας στην οποία αποθηκεύεται η πληροφορία για τα δικαιώματα του χρήστη που συνδέεται στο σύστημα.

`User user`: πεδίο, αποθηκεύονται οι πληροφορίες των χρηστών που επιχειρούν να συνδεθούν στον εξυπηρετητή για να γίνει επαλήθευση τους. ,

`GameTable table`: πεδίο, περιέχει τις πληροφορίες του τραπεζιού.

`List<Player> players`: πεδίο, η λίστα με όλους τους παίκτες που έχουν συνδεθεί επιτυχημένα.

`TimeSpan timetopass`: πεδίο, ο χρόνος που πρέπει να περάσει από το τελευταίο μοίρασμα κάρτας μέχρι το επόμενο.

`DateTime time`: πεδίο, αποθηκεύεται η ώρα του συστήματος.

`Timer aTimer`: πεδίο, ο χρόνος μετά το πέρας του οποίου θα πυροδοτείται το `AutoFoldEvent`.

`bool turnStart`: πεδίο, δυαδική μεταβλητή αλήθειας που αντιπροσωπεύει την αρχή του γύρου.

`int playerToDeal`: πεδίο, δείκτης στην θέση του παίκτη που θα μοιραστεί η επομένη κάρτα.

`int activePlayers`: πεδίο, οι ενεργοί παίκτες για τη συγκεκριμένη στιγμή της παρτίδας.

`int dealerPos`: πεδίο, η θέση του τραπεζιού στην οποία βρίσκεται ο παίκτης που μοιράζει τις κάρτες.

`bool firstCard`: πεδίο, δυαδική μεταβλητή αλήθειας που δείχνει αν έχει μοιραστεί η πρώτη κάρτα σε όλους τους παίκτες.

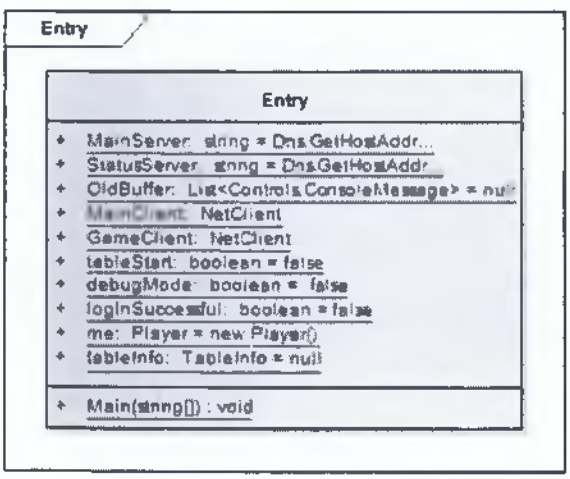
`bool dealCard`: πεδίο, δυαδική μεταβλητή αλήθειας, απαντά αν πρέπει να μοιραστεί κάρτα.
`bool startBet`: πεδίο, δυαδική μεταβλητή αλήθειας, απαντά αν πρέπει να αρχίσουν τα πονταρίσματα.
`bool showCommon`: πεδίο, δυαδική μεταβλητή αλήθειας, απαντά αν πρέπει να εμφανιστούν οι κοινές κάρτες.
`bool findWinner`: πεδίο, δυαδική μεταβλητή αλήθειας, απαντά αν πρέπει να αναζητηθεί ο νικητής.
`TurnStep common`: πεδίο, σε ποιά φάση «ανοίγματος» των κοινών φύλλων βρισκόμαστε.
`int playerToBet`: πεδίο, ο παίκτης που θα πρέπει να ποντάρει.
`int maxBet`: πεδίο, η τιμή του μέγιστου πονταρίσματος από παίκτη.
`int maxBetPos`: πεδίο, η θέση του παίκτη με το μέγιστο ποντάρισμα.
`int maxRaise`: πεδίο, το μέγιστο ποντάρισμα που επιτρέπεται στον συγκεκριμένο παίκτη.
`Deck deck`: πεδίο, η τράπουλα του παιχνιδιού.
`List<Card> commCards`: πεδίο, οι κοινές κάρτες.
`List<Seat> seats`: πεδίο, αντίγραφο από τις θέσεις του τραπεζιού.
`Main`: μέθοδος, η κεντρική μέθοδος μέσα στη οποία εκτελούνται οι εντολές και γίνονται κλήσεις προς τις υπόλοιπες μεθόδους.
`waitForAnswer`: μέθοδος, αναμονή για απάντηση από τον εξυπηρετητή.
`SitOut`: μέθοδος, εκτελεί τη διαδικασία αποχώρησης του αιτούντος παίκτη από τη θέση του.
`deal`: μέθοδος, μοιράζει μια κάρτα σε κάθε παίκτη που συμμετέχει στην παρτίδα.
`setDealer`: μέθοδος, βρίσκει τη θέση που παίκτη που θα μοιράσει τις κάρτες για αυτή την παρτίδα.
`setBlinds`: μέθοδος, τοποθετεί τα τυφλά πονταρίσματα για τους παίκτες που απαιτείται.
`askForBet`: μέθοδος, δίνει εντολή σε συγκεκριμένο παίκτη προκειμένου να επιλέξει την κίνηση που επιθυμεί για την συγκεκριμένη φάση της παρτίδας.
`SetBet`: μέθοδος, στέλνει τα δεδομένα από τις επιλογές πονταρίσματος ενός παίκτη στους υπόλοιπους παίκτες που βρίσκονται στο συγκεκριμένο τραπέζι.
`ShowCommon`: μέθοδος, εμφανίζει τις κοινές κάρτες.
`ShowCards`: μέθοδος, διαχειρίζεται την διαδικασία εμφάνισης των καρτών ενός η περισσότερων παικτών στους υπόλοιπους παίκτες.
`MaxRaise`: μέθοδος, υπολογίζει το μέγιστο ποντάρισμα για τον παίκτη μιας δεδομένης θέσης.
`MinBet`: μέθοδος, υπολογίζει το ελάχιστο ποντάρισμα για τον παίκτη μιας δεδομένης θέσης.
`BetManager`: μέθοδος, διαχειρίζεται το ποντάρισμα των παικτών στην πλευρά του εξυπηρετητή.
`DealPots`: μέθοδος, μοιράζει τα εικονικά χρήματα στους νικητές της παρτίδας.

`FindBestFive`: μέθοδος, εκτελεί τον αλγόριθμο για την αναζήτηση της καλύτερης πεντάδας για κάθε παίκτη.
`TurnReset`: μέθοδος, αρχικοποίηση των απαραίτητων μεταβλητών και εκκίνηση νέας παρτίδας.
`AutoFoldEvent`: μέθοδος, απορρίπτει τις κάρτες του παίκτη μετά από συγκεκριμένο χρονικό διάστημα και τον θέτει εκτός παιχνιδιού για την παρούσα παρτίδα.

3.4 Εφαρμογή Πελάτη

Στις παρακάτω ενότητες παρουσιάζονται οι κλάσεις που υλοποιήθηκαν στο πλαίσιο ανάπτυξης της εφαρμογής πελάτη. Οι κλάσεις χρησιμοποιούνται για την εμφάνιση των γραφικών και τη διαχείριση των κανόνων του παιχνιδιού.

3.4.1 Entry

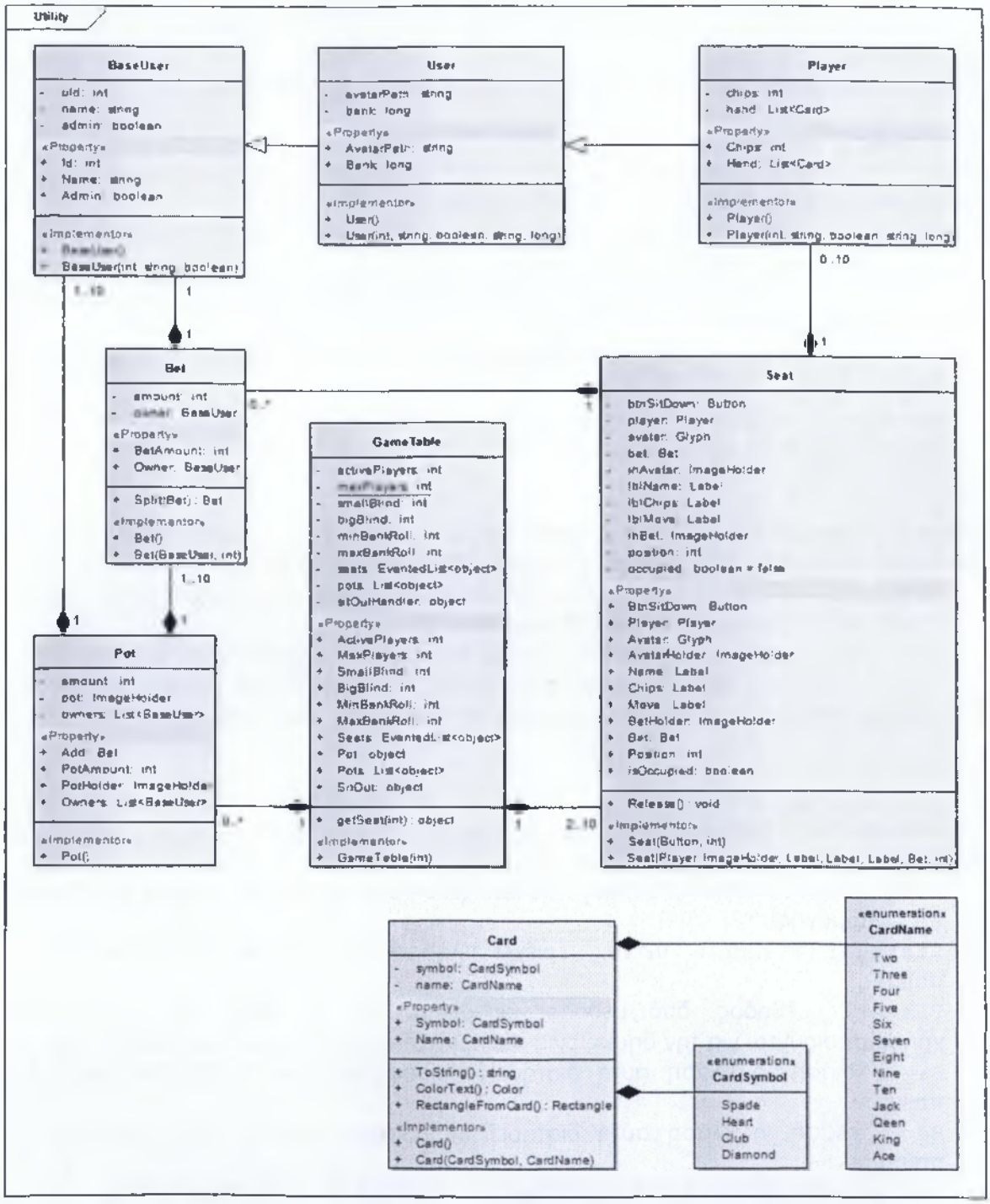


Σχήμα 3.4-1: Κλάση **Entry**

`Entry`
κλάση, είναι η κεντρική κλάση μέσα από την οποία γίνεται η εκκίνηση όλων των παραθύρων της εφαρμογής πελάτη. Κάθε παράθυρο που εκκινείται, εκτελείται σε αυτόνομο νήμα εκτέλεσης.
`string MainServer`: πεδίο, η ηλεκτρονική διεύθυνση που απαιτείται για την σύνδεση στον κεντρικό εξυπηρετητή.
`string StatusServer`: πεδίο, η ηλεκτρονική διεύθυνση που απαιτείται για την σύνδεση στον εξυπηρετητή κατάστασης
`List<ConsoleMessage> OldBuffer`: πεδίο, η λίστα στην οποία αποθηκεύονται όλα τα εισερχόμενα μηνύματα επικοινωνίας προς την εφαρμογή πελάτη.

`NetClient MainClient`: πεδίο, είναι η βασική κλάση της βιβλιοθήκης Lidgren network και περιέχει όλες τις πληροφορίες και τις λειτουργίες που αφορούν στην εφαρμογή πελάτη.
`NetClient GameClient`: πεδίο, είναι βασική κλάση της βιβλιοθήκης Lidgren network και περιέχει όλες τις πληροφορίες και τις λειτουργίες που αφορούν στην εφαρμογή πελάτη.
`bool tableStart`: πεδίο, δυαδική μεταβλητή αλήθειας, απαντά αν η εφαρμογή βρίσκεται στο παράθυρο του τραπεζιού.
`bool debugMode`: πεδίο, δυαδική μεταβλητή αλήθειας, απαντά αν η εφαρμογή είναι σε κατάσταση απασφαλμάτωσης.
`bool logInSuccessful`: πεδίο, δυαδική μεταβλητή αλήθειας, απαντά εάν υπάρχει επιτυχής σύνδεση του χρήστη στην εφαρμογή.
`Player me`: πεδίο, οι πληροφορίες του παίκτη που έχει συνδεθεί επιτυχημένα στην εφαρμογή.
`TableInfo tableInfo`: πεδίο, τα δεδομένα της κλάσης του τραπεζιού στο οποίο έχει συνδεθεί ο χρήστης.
`Main`: μέθοδος, η κεντρική μέθοδος του προγράμματος μέσα από την οποία γίνονται οι κλήσεις προς τις υπόλοιπες κλάσεις του συστήματος μας.

3.4.2 Utility



Σχήμα 3.4-2: Κλάση Utility

CardName
κλάση απαρίθμησης, οι ονομασίες των καρτών της τράπουλας.

CardSymbol
κλάση απαρίθμησης, τα τέσσερα σύμβολα των καρτών της τράπουλα.

BaseUser
κλάση, απεικονίζει τις πληροφορίες που αφορούν στους χρήστες της εφαρμογής σε αφαιρετικό επίπεδο. Διαθέτει τα παρακάτω πεδία και μεθόδους.

`int uId`: πεδίο, το μοναδικό αναγνωριστικό κάθε αντικειμένου της κλάσης.
`string name`: πεδίο, το όνομα του χρήστη.
`NetConnection connection`: πεδίο, τα στοιχεία σύνδεσης στην εφαρμογή πελάτη του χρήστη.
`boolean admin`: πεδίο, τα δικαιώματα πρόσβασης του χρήστη στις λειτουργίες της εφαρμογής.
`BaseUser`: μέθοδος, δυο μέθοδοι που φέρουν το ίδιο όνομα με την κλάση και χρησιμοποιούνται για την δημιουργία και αρχικοποίηση των αντικειμένων αυτής.

User
κλάση, κληρονομεί από την κλάση BaseUser και την επεκτείνει προσθέτοντας πεδία για τις πληροφορίες που απαιτούνται μετά την σύνδεση του χρήστη στο σύστημα.

`string avatarPath`: πεδίο, η διαδρομή που αντιστοιχεί στην εικόνα προσωπείου του χρήστη μέσα στο σύστημα.
`long bank`: πεδίο, οι συνολικές μάρκες που διαθέτει ο χρήστης στην κατοχή του.
`User`: μέθοδος, δυο μέθοδοι που έχουν το ίδιο όνομα με την κλάση και χρησιμοποιούνται για την δημιουργία και αρχικοποίηση των αντικειμένων αυτής.

Player
κλάση, κληρονομεί από την κλάση User και την επεκτείνει προσθέτοντας πεδία για της πληροφορίες που απαιτούνται προκειμένου να μπορεί ένας χρήστης να συμμετέχει στα τραπέζια.

`long chips`: πεδίο, οι μάρκες που κατέχει ο χρήστης βρισκόμενος σε ένα τραπέζι της εφαρμογής.
`List<Card> hand`: πεδίο, περιέχει πληροφορίες για τις κάρτες που έχει ο παίκτης.
`Player`: μέθοδος, δυο μέθοδοι με όνομα ίδιο με αυτό της κλάσης που χρησιμοποιούνται για την δημιουργία και αρχικοποίηση των αντικειμένων αυτής.
`Bet`: κλάση, η κλάση αυτή διατηρεί πληροφορίες για τα πονταρίσματα των παικτών.
`Bet`: κλάση, η κλάση αυτή διατηρεί πληροφορίες για τα πονταρίσματα των παικτών.
`int amount`: πεδίο, η αριθμητική τιμή του συγκεκριμένου πονταρίσματος.
`BaseUser owner`: πεδίο, οι πληροφορίες του παίκτη στον οποίο ανήκει το ποντάρισμα.

`Split`: μέθοδος, η μέθοδος αυτή χρησιμοποιείται για την διαίρεση του πονταρίσματος, στην περίπτωση που πρέπει να δημιουργηθεί μια πλαϊνή στοίβα πονταρίσματος (Side Pot).

`Bet`: μέθοδος, δυο μέθοδοι με όνομα ίδιο με αυτό της κλάσης που χρησιμοποιούνται για την δημιουργία και αρχικοποίηση των αντικειμένων αυτής.

`Pot`

κλάση, χρησιμοποιείται για την διατήρηση των πληροφοριών σχετικά με τα πονταρίσματα που γίνονται σε κάθε παρτίδα. Σε ένα τραπέζι μπορούν να υπάρχουν πολλά διαφορετικά στιγμιότυπα της κλάσης.

`int amount`: πεδίο, το άθροισμα των μαρκών που περιέχονται στην συγκεκριμένη στοίβα πονταρίσματος.

`List<BaseUser> owners`: πεδίο, όλοι οι χρήστες που έχουν συνεισφέρει στην στοίβα πονταρίσματος, και επομένως έχουν δικαίωμα απόκτησης της.

`ImageHolder pot`: πεδίο, η γραφική απεικόνιση στο παράθυρο του πόσου της στοίβας πονταρίσματος.

`Bet Add`: ιδιότητα, προσθέτει ένα νέο ποντάρισμα στη στοίβα πονταρισμάτων.

`Pot`: μέθοδος, μέθοδος για την δημιουργία αντικειμένων της συγκεκριμένης κλάσης.

`Card`

κλάση, αντιπροσωπεύει στο πρόγραμμα μας της κάρτες μιας τράπουλας όπως αυτές παρουσιάζονται στον πραγματικό κόσμο.

`CardSymbol symbol`: πεδίο, ένα από τα τέσσερα πιθανά σύμβολα που μπορεί να έχει μια κάρτα.

`CardName name`: πεδίο, το όνομα της κάρτας σύμφωνα με την ονομασία που έχουν στον πραγματικό κόσμο.

`ToString`: μέθοδος, επιστρέφει την ονομασία και το σύμβολο της κάρτας.

`Card`: μέθοδος, δυο μέθοδοι με όνομα ίδιο με αυτό της κλάσης και σκοπό την δημιουργία και αρχικοποίηση των αντικειμένων αυτής.

`Seat`

κλάση, συμβολίζει την θέση που κατέχει ο παίκτης όπως αυτή παρουσιάζεται στα τραπέζια παιχνιδιών.

`Button btnSitDown`: πεδίο, είναι το κουμπί μέσω του οποίου ο παίκτης κατοχυρώνει τη συγκεκριμένη θέση του τραπεζιού.

`Player player`: πεδίο, τα στοιχεία του παίκτη ο οποίος κατέχει τη θέση τη δεδομένη χρονική στιγμή.

`Glyph avatar`: πεδίο, περιέχει τις πληροφορίες για το προσωπείο του παίκτη που κατέχει τη συγκεκριμένη θέση στο τραπέζι.

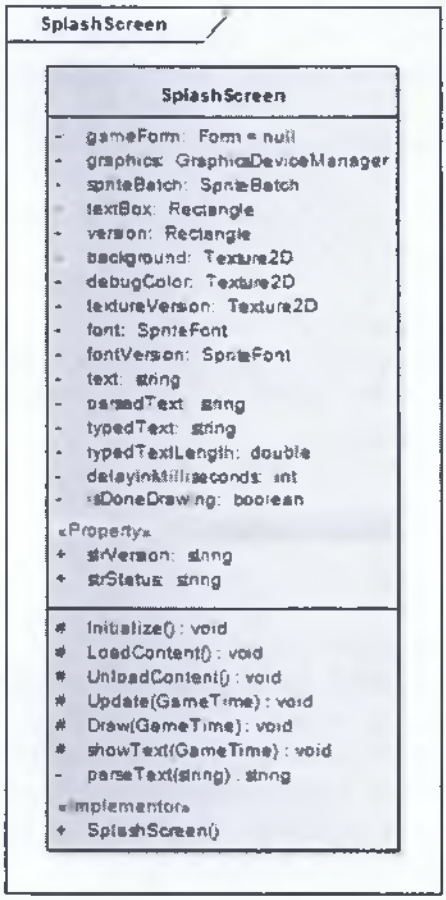
`Bet bet`: πεδίο, σε αυτή τη μεταβλητή τοποθετούνται τα πονταρίσματα των παικτών μέχρι να μπουν στη στοίβα πονταρίσματος.

`ImageHolder ihAvatar`: πεδίο, η γραφική απεικόνιση του προσωπείου του παίκτη στην εφαρμογή.

`Label lblName`: πεδίο, η γραφική απεικόνιση του ονόματος του παίκτη που έχει τη θέση.
`Label lblChips`: πεδίο, η γραφική απεικόνιση των εικονικών χρημάτων που διαθέτει ο παίκτης.
`Label lblMove`: πεδίο, η γραφική απεικόνιση της επιλογής του παίκτη κατά την διάρκεια του ποντάρισματος.
`ImageHolder ihBet`: πεδίο, η γραφική απεικόνιση του ποσού των εικονικών χρημάτων που ποντάρει ο παίκτης.
`int position`: πεδίο, ο αριθμός της θέσης στο τραπέζι όπως αυτή ορίζεται από το σύστημα μας.
`bool occupied`: πεδίο, δυαδική μεταβλητή αλήθειας μέσω της οποίας ενημερωνόμαστε για το αν είναι κατειλημμένη ή ελεύθερη η συγκεκριμένη θέση.
`Release`: μέθοδος, «απελευθερώνει» τη συγκεκριμένη θέση στην περίπτωση που είναι κατειλημμένη.
`Seat`: μέθοδος, δυο μέθοδοι με όνομα ίδιο με αυτό της κλάσης που χρησιμοποιούνται για την δημιουργία και αρχικοποίηση των αντικειμένων αυτής.

`GameTable`
κλάση, είναι η αναπαράσταση του τραπεζιού στο οποίο παίζεται το παιχνίδι.
`int activePlayers`: πεδίο, οι «ενεργοί» παίκτες στο συγκεκριμένο τραπέζι.
`int maxPlayers`: πεδίο, ο μέγιστος αριθμός των παικτών που μπορούν να συμμετάσχουν στο συγκεκριμένο τραπέζι.
`int smallBlind`: πεδίο, το μικρό τυφλό ποντάρισμα όπως αυτό είναι ορισμένο από τους κανόνες του παιχνιδιού.
`int bigBlind`: πεδίο, το μεγάλο τυφλό ποντάρισμα όπως αυτό είναι ορισμένο από τους κανόνες του παιχνιδιού.
`int minBankRoll`: πεδίο, το ελάχιστο ποσό που απαιτείται για την είσοδο του παίκτη στο τραπέζι.
`int maxBankRoll`: πεδίο, το μέγιστο ποσό με το οποίο μπορεί να εισέλθει ο παίκτης στο τραπέζι.
`EventedList<object> seats`: πεδίο, η λίστα με τις θέσεις του τραπεζιού. Κάθε φορά που είτε προστίθεται ένα νέο πεδίο στην λίστα είτε αφαιρείται κάποιο πυροδοτείται η κατάλληλη μέθοδος για το χειρισμό του γεγονότος.
`List<object> pots`: πεδίο, η λίστα με τις στοίβες ποντάρισμάτων του τραπεζιού.
`object sitOutHandler`: πεδίο, η γραφική μέθοδος με την οποία θα γίνεται η αποχώρηση του παίκτη από την συγκεκριμένη θέση του τραπεζιού.
`GetSeat`: μέθοδος, επιστρέφει την αιτούμενη βάση αριθμού θέση.
`GameTable`: μέθοδος, μέθοδος για τη δημιουργία και αρχικοποίηση αντικειμένων της κλάσης.

3.4.3 SplashScreen



Σχήμα 3.4-3: Κλάση οθόνης υποδοχής

`SplashScreen`
κλάση, χρησιμοποιείται για την δημιουργία παράθυρου κατά την εκκίνηση της εφαρμογής στο οποίο εμφανίζονται πληροφορίες για την εργασία στα πλαίσια της οποίας δημιουργήθηκε η συγκεκριμένη εφαρμογή.

`Form gameForm`: πεδίο, αντιπροσωπεύει ένα παράθυρο ή ένα πλαίσιο διαλόγου το οποίο δημιουργεί το γραφικό περιβάλλον χρήστη της εφαρμογής.

`GraphicsDeviceManager`: πεδίο, χειρίζεται τις ρυθμίσεις και την διαχείριση της συσκευής των γραφικών.

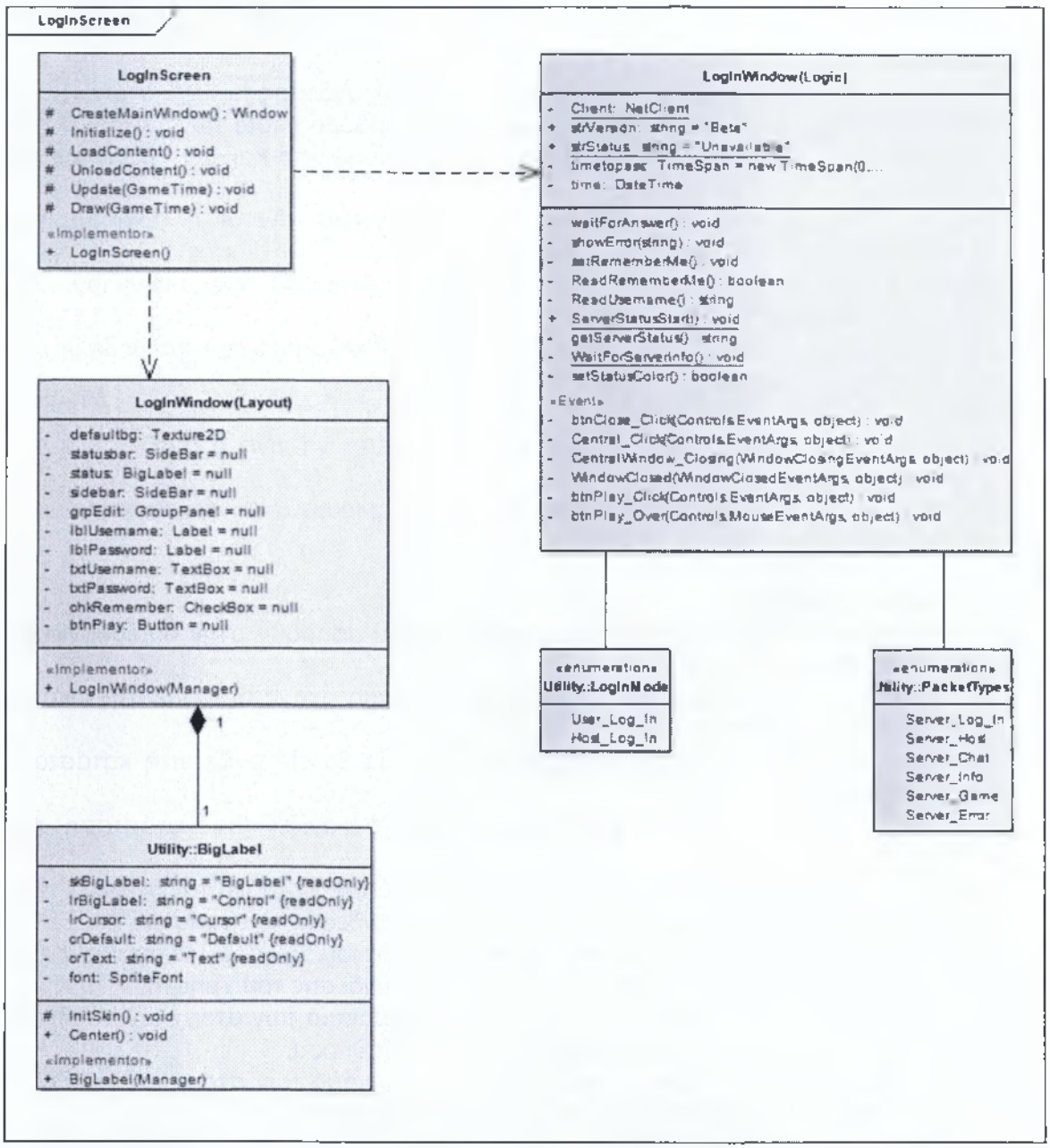
`SpriteBatch spriteBatch`: πεδίο, επιτρέπει σε ένα σύνολο από στοιχεία να αναπαρασταθούν με τις ίδιες ρυθμίσεις.

`Rectangle textBox`: πεδίο, ορίζει ένα παραλληλόγραμμο μέσα στο οποίο θα αποτυπωθεί το κείμενο με το σκοπό της εφαρμογής.

`Rectangle version`: πεδίο, ορίζει ένα παραλληλόγραμμο μέσα στο οποίο θα εμφανίζεται η έκδοση της εφαρμογής.

`Texture2D background`: πεδίο, αντιπροσωπεύει ένα δυσδιάστατο πλέγμα από εικονοστοιχεία της εικόνας παρασκηνίου.
`Texture2D debugColor`: πεδίο, δυσδιάστατο πλέγμα από εικονοστοιχεία τα οποία χρησιμοποιούνται για την διαδικασία απασφαλμάτωσης.
`Texture2D textureVersion`: πεδίο, δυσδιάστατο πλέγμα από εικονοστοιχεία της εικόνας παρασκηνίου του παραλληλόγραμμου στο οποίο εμφανίζεται η έκδοση της εφαρμογής.
`SpiteFont font`: πεδίο, αντιπροσωπεύει τις ιδιότητες της γραμματοσειράς του κειμένου με το σκοπό της εφαρμογής.
`SpiteFont fontVersion`: πεδίο, αντιπροσωπεύει τις ιδιότητες της γραμματοσειράς του κειμένου της έκδοσης.
`string text`: πεδίο, το κείμενο με τον σκοπό της εφαρμογής.
`string parsedText`: πεδίο, μεταβλητή που χρησιμοποιείται για δημιουργία εφέ σταδιακής εμφάνισης του κειμένου.
`string typedText`: πεδίο, μεταβλητή που χρησιμοποιείται για δημιουργία εφέ σταδιακής εμφάνισης του κειμένου.
`double typedTextLength`: πεδίο, μεταβλητή που χρησιμοποιείται για δημιουργία εφέ σταδιακής εμφάνισης του κειμένου.
`int delayInMilliseconds`: πεδίο, μεταβλητή που χρησιμοποιείται για δημιουργία εφέ σταδιακής εμφάνισης του κειμένου.
`bool isDoneDrawing`: πεδίο, δυαδικη μεταβλητή αλήθειας που χρησιμοποιείται για δημιουργία εφέ σταδιακής εμφάνισης του κειμένου.
`string strVersion`: πεδίο, η έκδοση της εφαρμογής πελάτη.
`string strStatus`: πεδίο, η κατάσταση των εξυπηρετητών.
`Initialize`: μέθοδος, η μέθοδος μέσα στην οποία γίνεται η αρχικοποίηση στοιχείων πριν από την έναρξη της εφαρμογής.
`LoadContent`: μέθοδος, καλείται μια φορά στην εκτέλεση του παιχνιδιού, εδώ γίνεται η «φόρτωση» των στοιχείων της εφαρμογής.
`UnloadContent`: μέθοδος, καλείται μια φορά στην εκτέλεση του παιχνιδιού, εδώ γίνεται η αποδέσμευση των πόρων της εφαρμογής.
`Update`: μέθοδος, εκτελεί την «λογική» του παιχνιδιού.
`Draw`: μέθοδος, αυτή η μέθοδος καλείται όταν πρέπει να γίνει ο σχεδιασμός των γραφικών του παιχνιδιού.
`showText`: μέθοδος, εμφανίζει το κείμενο προσδίδοντας εφέ σταδιακής εμφάνισης.
`parseText`: μέθοδος, βοηθητική μέθοδος για την δημιουργία εφέ σταδιακής εμφάνισης του κειμένου.
`SplashScreen`: μέθοδος, χρησιμοποιείται για την δημιουργία και αρχικοποίηση των αντικειμένων της κλάσης.

3.4.4 LogInScreen



Σχήμα 3.4-4: Διάγραμμα κλάσεων της οθόνης σύνδεσης

`LogInMode`
κλάση απαρίθμησης, χρησιμοποιείται για τον διαχωρισμό των χρηστών που συνδέονται στον εξυπηρετητή.

`PacketTypes`: πεδίο, σκοπός της είναι η «περιτύλιξη» των μηνυμάτων που λαμβάνονται και αποστέλλονται από και προς τον εξυπηρετητή.

`LogInScreen`

κλάση, η κεντρική κλάση που διαχειρίζεται το παράθυρο σύνδεσης των χρηστών.

`CreateMainWindow`: μέθοδος, δημιουργεί το παράθυρο μέσω των κλάσεων που είναι υπεύθυνες για την υλοποίηση τόσο της εμφάνισης όσο και της «λογικής» της κλάσης.

`Initialize`: μέθοδος, η μέθοδος μέσα στην οποία γίνεται η αρχικοποίηση στοιχείων πριν από την έναρξη της εφαρμογής.

`LoadContent`: μέθοδος, καλείται μια φορά στην εκτέλεση του παιχνιδιού, εδώ γίνεται η «φόρτωση» των στοιχείων της εφαρμογής.

`UnloadContent`: μέθοδος, καλείται μια φορά στην εκτέλεση του παιχνιδιού, εδώ γίνεται η αποδέσμευση των πόρων της εφαρμογής.

`Update`: μέθοδος, εκτελεί την «λογική» του παιχνιδιού.

`Draw`: μέθοδος, αυτή η μέθοδος καλείται όταν πρέπει να γίνει ο σχεδιασμός των γραφικών του παιχνιδιού.

`LogInScreen`: μέθοδος, χρησιμοποιείται για την δημιουργία και αρχικοποίηση των αντικειμένων της κλάσης.

`LogInWindow (Layout)`

κλάση, εδώ βρίσκονται όλα τα πεδία και οι μέθοδοι που αφορούν στην δημιουργία και στην εμφάνιση του παράθυρου.

`Texture2D defaultbg`: πεδίο, δυσδιάστατο πλέγμα εικονοστοιχείων της εικόνας παρασκηνίου του παράθυρου.

`SideBar statusbar`: πεδίο, ο χώρος στον οποίο θα εμφανίζεται η κατάσταση των εξυπηρετητών.

`BigLabel Status`: πεδίο, η ετικέτα η οποία εμφανίζει την κατάσταση των εξυπηρετητών.

`SideBar sidebar`: πεδίο, το «κουτί» που θα φιλοξενήσει τα στοιχεία που χρησιμοποιούνται για την εισαγωγή των στοιχείων σύνδεσης του χρήστη.

`GroupPanel grpEdit`: πεδίο, το «κουτί» που θα φιλοξενήσει τα στοιχεία που χρησιμοποιούνται για την εισαγωγή των στοιχείων σύνδεσης του χρήστη.

`Label lblUsername`: πεδίο, ετικέτα για των διαχωρισμό των στοιχείων σύνδεσης που θα εισάγονται σε κάθε γραφικό στοιχειό του παράθυρου.

`Label lblPassword`: πεδίο, ετικέτα για το διαχωρισμό των στοιχείων σύνδεσης που θα εισάγονται σε κάθε πλαίσιο κειμένου του παράθυρου.

`TextBox txtUsername`: πεδίο, το πλαίσιο κειμένου μέσα στο οποίο θα εισάγεται το ψευδώνυμο του χρήστη που επιθυμεί να συνδεθεί.

`TextBox txtPassword`: πεδίο, το πλαίσιο κειμένου με στο οποίο θα εισάγεται το συνθηματικό του χρήστη που επιθυμεί να συνδεθεί.

`CheckBox chkRemember`: πεδίο, πλαίσιο ελέγχου το οποίο δίνει τη δυνατότητα στο χρήστη για την απομνημόνευση του ψευδώνυμου που έχει εισάγει στο αντίστοιχο πλαίσιο κειμένου, για μελλοντική χρήση αυτού.

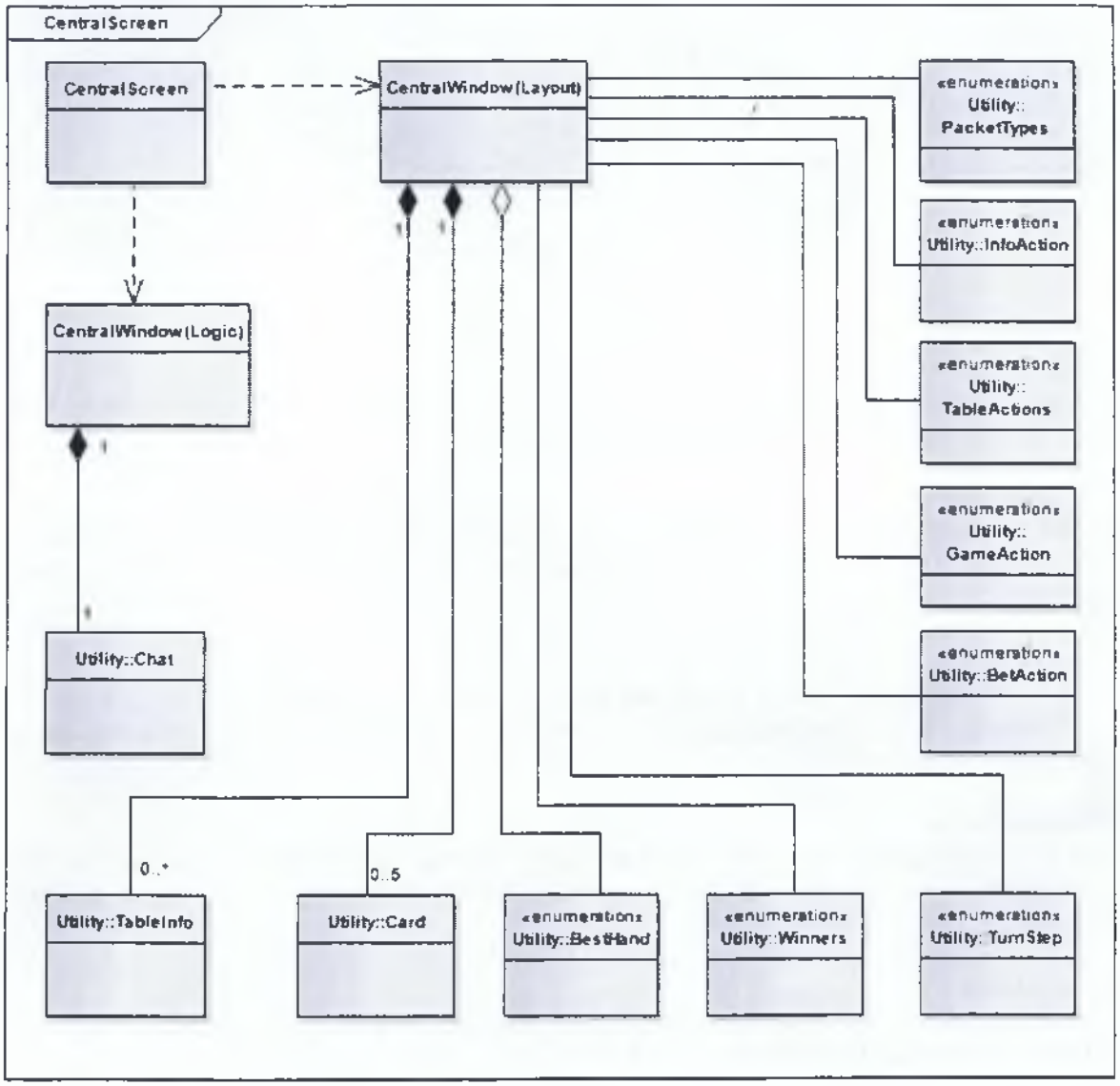
`Button btnPlay`: πεδίο, το κουμπί αίτηση σύνδεσης στον εξυπηρετητή και μετάβαση στο κεντρικό παράθυρο της εφαρμογής.
`LogInWindow`: μέθοδος, χρησιμοποιείται για την δημιουργία και την αρχικοποίηση των αντικειμένων της κλάσης.

`LogInWindow (Logic)`
κλάση, σε αυτήν την κλάση υπάρχουν όλα τα αντικείμενα και οι μέθοδοι που χρησιμοποιούνται για την υλοποίηση της «λογικής» του προγράμματος.
`NetClient Client`: πεδίο, είναι βασική κλάση της βιβλιοθήκης Lidgren network και περιέχει όλες τις πληροφορίες και τις λειτουργίες που αφορούν στην εφαρμογή πελάτη.
`string strVersion`: πεδίο, συμβολοσειρά με την έκδοση της εφαρμογής πελάτη.
`string strStatus`: πεδίο, συμβολοσειρά με την κατάσταση των εξυπηρετητών.
`TimeSpan timetopass`: πεδίο, ο χρόνος μετά το πέρας του οποίου θα γίνεται εκ νέου έλεγχος για την κατάσταση των εξυπηρετητών.
`DateTime time`: πεδίο, η ώρα του συστήματος της εφαρμογής πελάτη.
`waitForAnswer`: μέθοδος, είναι η μέθοδος η οποία θα «διαβάσει» τα δεδομένα του εξυπηρετητή μετά από μια αίτηση σύνδεσης σε αυτόν.
`showError`: μέθοδος, εμφανίζει μέσα σε πλαίσια διαλόγου τα σφάλματα και τις προειδοποιήσεις του αποστέλλονται από τον εξυπηρετητή.
`setRememberMe`: μέθοδος, διαχειρίζεται τις διαδικασίες για την υλοποίηση της λειτουργίας απομνημόνευσης του ψευδώνυμου του χρήστη που συνδέεται.
`ReadRememberMe`: μέθοδος, διαβάζει τα δεδομένα που αφορούν στην λειτουργία απομνημόνευσης του ψευδώνυμου του χρήστη.
`ReadUsername`: μέθοδος, διαβάζει το ψευδώνυμο του χρήστη από το αρχείο ρυθμίσεων και το εισάγει στο αντίστοιχο πλαίσιο κειμένου του παράθυρου.
`ServerStatusStart`: μέθοδος, εκκινεί την διεργασία η οποία «διαβάζει» αυτόματα την κατάσταση των εξυπηρετητών και ενημερώνει την αντίστοιχη λεζάντα στο παράθυρο της εφαρμογής.
`getServerStatus`: μέθοδος, η διεργασία η οποία «διαβάζει» αυτόματα την κατάσταση των εξυπηρετητών και ενημερώνει την αντίστοιχη λεζάντα στο παράθυρο της εφαρμογής.
`WaitForServerInfo`: μέθοδος, είναι η μέθοδος η οποία περιμένει και διαβάζει τα δεδομένα που αφορούν στην κατάσταση των εξυπηρετητών.
`SetStatusColor`: μέθοδος, αλλάζει το χρώμα του κειμένου της λεζάντας η οποία εμφανίζει την κατάσταση των εξυπηρετητών στο χρήστη.
`btnClose_Click`: μέθοδος, γεγονός που πυροδοτείται μετά το πάτημα του κουμπιού κλεισίματος του παράθυρου.
`Central_Click`: μέθοδος, γεγονός που πυροδοτείται προκειμένου να τερματίσει το παράθυρο το οποίο πυροδοτεί το γεγονός.
`CentralWindow_Closing`: μέθοδος, γεγονός που πυροδοτείται κατά την διάρκεια του τερματισμού του κεντρικού παράθυρου.

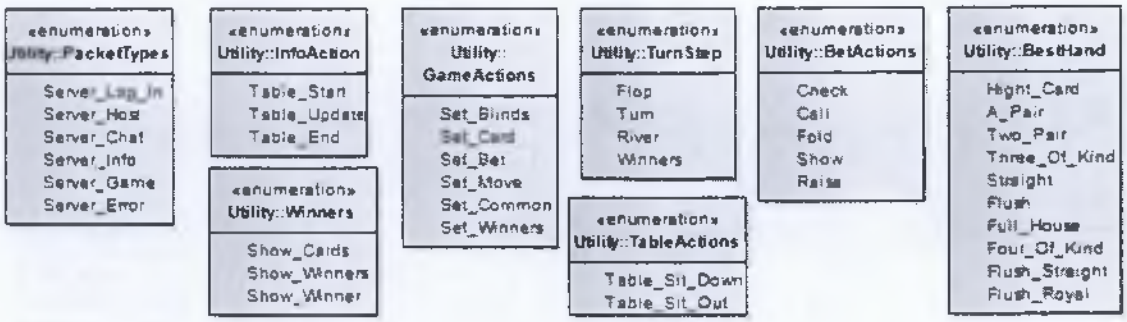
`WindowClosed`: μέθοδος, γεγονός που πυροδοτείται μετά το κλείσιμο του παράθυρου.
`btnPlay_Click`: μέθοδος, γεγονός που πυροδοτείται όταν πιέζεται το κουμπί με όνομα `Play`.
`btnPlay_Over`: μέθοδος, γεγονός που πυροδοτείται όταν ο κέρσορας του ποντικιού βρίσκεται επάνω από το κουμπί με όνομα Play.

`BigLabel`
κλάση, κληρονομεί από την κλάση Label του Neoforce και την επεκτείνει αλλάζοντας το μέγεθος της γραμματοσειράς της ετικέτας.
`string skBigLabel`: πεδίο, συμβολοσειρά που χρησιμοποιείται για την φόρτωση δεδομένων από τα αρχεία εμφάνισης.
`string lrBigLabel`: πεδίο, συμβολοσειρά που χρησιμοποιείται για την φόρτωση δεδομένων από τα αρχεία εμφάνισης.
`string lrCursor`: πεδίο, συμβολοσειρά που χρησιμοποιείται για την φόρτωση δεδομένων από τα αρχεία εμφάνισης.
`string crDefault`: πεδίο, συμβολοσειρά που χρησιμοποιείται για την φόρτωση δεδομένων από τα αρχεία εμφάνισης.
`string crText`: πεδίο, συμβολοσειρά που χρησιμοποιείται για την φόρτωση δεδομένων από τα αρχεία εμφάνισης.
`SpiteFont font`: πεδίο, τα δεδομένα του στοιχείου της γραμματοσειράς.
`InitSkin`: μέθοδος, κάνει αρχικοποίηση των στοιχείων που συνδέονται με τα αρχεία εμφάνισης.
`Center`: μέθοδος, τοποθετεί το συγκεκριμένο στοιχειό στο μέσο του γονικού του στοιχείου.
`BigLabel`: μέθοδος, χρησιμοποιείται για την δημιουργία και αρχικοποίηση των αντικειμένων της κλάσης.

3.4.5 CentralScreen



Σχήμα 3.4-5: Συνοπτικό διάγραμμα κλάσεων κεντρικής οθόνης



Σχήμα 3.4-6: Κλάσεις απαρίθμησης

PacketTypes
κλάση απαρίθμησης, σκοπός της είναι η «περιτύλιξη» των μηνυμάτων που λαμβάνονται και αποστέλλονται από τον εξυπηρετητή.

InfoAction
κλάση απαρίθμησης, με αυτήν την κλάση απαρίθμησης «μαρκάρονται» τα μηνύματα που αφορούν στη δημιουργία, στην ανανέωση, και στον τερματισμό των ενεργών τραπεζιών.

TableAction
κλάση απαρίθμησης, μέσω αυτής της κλάσης γίνεται η αναγνώριση των μηνυμάτων που περιέχουν πληροφορίες σχετικά με την συμμετοχή ενός παίκτη ή την αποχώρηση του από κάποιο τραπέζι.

GameActions
κλάση απαρίθμησης, με χρήση της παρούσας κλάσης διαχωρίζονται τα «πακέτα» που περιέχουν δεδομένα σχετικά με τις φάσεις (τυφλό ποντάρισμα, μοίρασμα φύλλων, ποντάρισμα, κ.τ.λ.) του παιχνιδιού.

BetActions
κλάση απαρίθμησης, απαριθμεί όλες τις δυνατές επιλογές και κινήσεις κάθε παίκτη με τα φύλλα που έχει στην κατοχή του.

TurnStep
κλάση απαρίθμησης, χρησιμοποιείται για την αρίθμηση των φάσεων της παρτίδας από το άνοιγμα των κοινών φύλλων μέχρι την ανακοίνωση του νικητή.

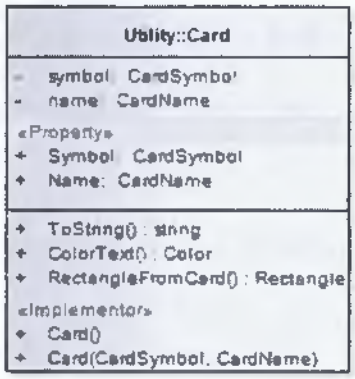
Winners
κλάση απαρίθμησης, με αυτήν την κλάση γίνεται η αναγνώριση των μηνυμάτων που περιέχουν πληροφορίες σχετικά με τις διαδικασίες ανακοίνωσης των νικητών.

BestHand

κλάση απαρίθμησης, περιέχει τις διακριτές ονομασίες των πιθανών σχημάτων που μπορούν να δημιουργηθούν από έναν παίκτη. Περιέχεται στην κλάση `BestHand`.

`Card`

κλάση, αντιπροσωπεύει στο πρόγραμμα μας τις κάρτες μιας τράπουλας όπως αυτές παρουσιάζονται στον πραγματικό κόσμο.



Σχήμα 3.4-7: Κλάση Card

`CardSymbol symbol`: πεδίο, ένα από τα τέσσερα πιθανά σύμβολα που μπορεί να έχει μια κάρτα.

`CardName name`: πεδίο, το όνομα της κάρτας σύμφωνα με την ονομασία που έχουν στον πραγματικό κόσμο.

`ToString`: μέθοδος, επιστρέφει την ονομασία και το σύμβολο της κάρτας.

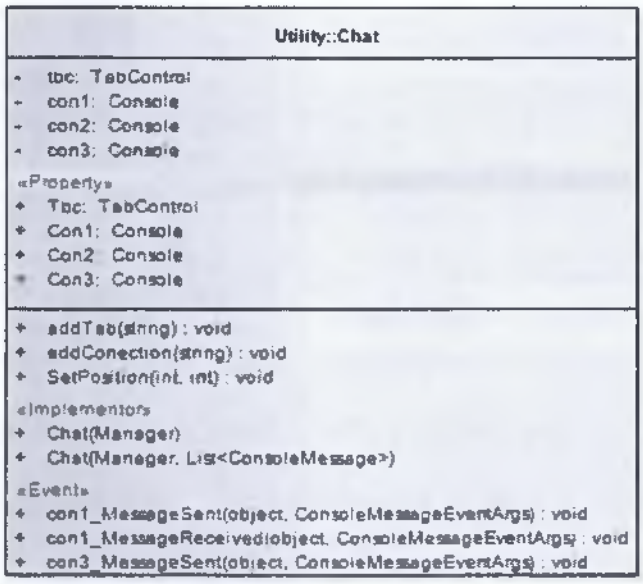
`ColorText`: μέθοδος, χρωματίζει το κείμενο του ονόματος της κάρτας σύμφωνα με το χρώμα που αυτή διαθέτει στον πραγματικό κόσμο.

`RectangleFromCard`: μέθοδος, επιστρέφει το παραλληλόγραμμο στο οποίο βρίσκεται η εικόνα της κάρτας μέσα από το αρχείο εικόνας.

`Card`: μέθοδος, δυο μέθοδοι με όνομα ίδιο με αυτό της κλάσης και σκοπό τη δημιουργία και αρχικοποίηση αντικειμένων αυτής.

`Chat`

κλάση, είναι η κλάση που αναπαριστά την γραφική κονσόλα που διαχειρίζεται τα μηνύματα επικοινωνίας.



Σχήμα 3.4-8: Κλάση Chat

`TabControl tbc`: πεδίο, δημιουργεί και διαχειρίζεται μια καρτέλα με αντικείμενα κονσόλας.

`Console con1`: πεδίο, κονσόλα που διαχειρίζεται τα μηνύματα που αφορούν σε ένα κανάλι επικοινωνίας.

`Console con2`: πεδίο, κονσόλα που διαχειρίζεται τα μηνύματα που αφορούν σε ένα κανάλι επικοινωνίας.

`Console con3`: πεδίο, κονσόλα που διαχειρίζεται τα μηνύματα που αφορούν σε ένα κανάλι επικοινωνίας.

`addTab`: πεδίο, δημιουργεί μια νέα καρτέλα.

`addConnection`: πεδίο, προσθέτει ένα νέο κανάλι επικοινωνίας σε ένα πλαίσιο

κονσόλας.

`SetPosition`: μέθοδος, θέτει τη θέση του στοιχείου μέσα στο παράθυρο.

`con1_MessageSent`: μέθοδος, γεγονός που πυροδοτείται όταν πληκτρολογείται ένα μήνυμα στο πρώτο κανάλι επικοινωνίας.

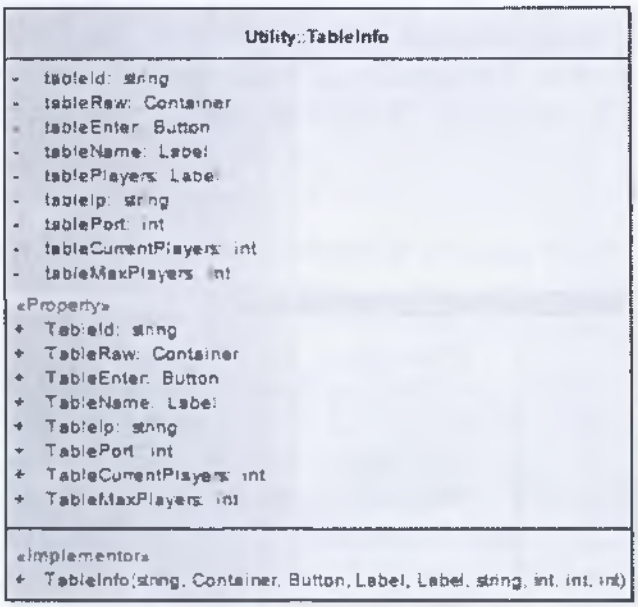
`con1_MessageReceived`: μέθοδος, γεγονός που πυροδοτείται όταν λαμβάνεται ένα μήνυμα στο πρώτο κανάλι επικοινωνίας.

`Con3_MessageSent`: μέθοδος, γεγονός που πυροδοτείται όταν πληκτρολογείται ένα μήνυμα στο τρίτο κανάλι επικοινωνίας.

`Chat`: μέθοδος, δυο μέθοδοι για τη δημιουργία και αρχικοποίηση των αντικειμένων της κλάσης.

`TableInfo`

κλάση, περιλαμβάνει πληροφορίες σχετικά με τα ενεργά τραπέζια του συστήματος. Αποτελείται από τα παρακάτω πεδία και μεθόδους:



Σχήμα 3.4-9: Κλάση TableInfo

`string tableId`: πεδίο, το μοναδικό αναγνωριστικό κάθε τραπεζιού.

`Container tableRaw`: πεδίο, είναι το «κουτί» που περιλαμβάνει τα γραφικά στοιχεία με τις πληροφορίες για τα ενεργά τραπέζια.

`Button tableEnter`: πεδίο, είναι το κουμπί μέσω του οποίου γίνεται η μετάβαση στο συγκεκριμένο τραπέζι.

`Label tablePlayers`: πεδίο, ετικέτα με των αριθμό των ενεργών και του συνόλου των παικτών που μπορούν να συμμετάσχουν στο τραπέζι.

`string tableIp`: πεδίο, η ηλεκτρονική διεύθυνση για την σύνδεση στον εξυπηρετητή παιχνιδιού που φιλοξενεί το συγκεκριμένο τραπέζι.

`int tablePort`: πεδίο, η «πόρτα» για την σύνδεση στον εξυπηρετητή παιχνιδιού που φιλοξενεί το συγκεκριμένο τραπέζι.

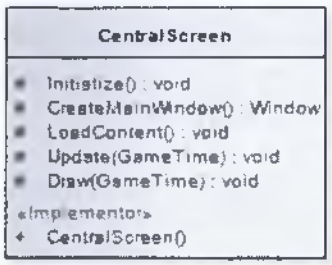
`int tableCurrentPlayers`: πεδίο, ο αριθμός των παικτών που βρίσκονται στο τραπέζι.

`int tableMaxPlayers`: πεδίο, ο μέγιστος αριθμός των παικτών που φιλοξενεί το συγκεκριμένο τραπέζι.

`TableInfo`: μέθοδος, χρησιμοποιείται για την δημιουργία αντικειμένων της συγκεκριμένης κλάσης.

`CentralScreen`

κλάση, η κεντρική κλάση που δημιουργεί και διαχειρίζεται το παράθυρο που εμφανίζεται με την σύνδεση των χρηστών. Το συγκεκριμένο παράθυρο χρησιμοποιείται και ως λόμπι `CreateMainWindow:` μέθοδος, δημιουργεί το παράθυρο μέσω των κλάσεων που είναι υπεύθυνες για την υλοποίηση τόσο της εμφάνισης όσο και της «λογικής» της κλάσης.



Σχήμα 3.4-10: Κλαση CentralScreen

`Initialize:` μέθοδος, η μέθοδος μέσα στην οποία γίνεται η αρχικοποίηση στοιχείων πριν από την έναρξη της εφαρμογής.

`LoadContent:` μέθοδος, καλείται μια φορά στην εκτέλεση του παιχνιδιού, εδώ γίνεται η «φόρτωση» των στοιχείων της εφαρμογής.

`UnloadContent:` μέθοδος, καλείται μια φορά στην εκτέλεση του παιχνιδιού, εδώ γίνεται η αποδέσμευση των πόρων της εφαρμογής.

`Update:` μέθοδος, εκτελεί την «λογική» του παιχνιδιού.

`Draw:` μέθοδος, αυτή η μέθοδος καλείται όταν πρέπει να γίνει ο σχεδιασμός των γραφικών του παιχνιδιού.

`CentralScreen:` μέθοδος, χρησιμοποιείται για την δημιουργία και αρχικοποίηση των αντικειμένων της κλάσης.

`CentralWindow (Layout)`

κλάση, εδώ βρίσκονται όλα τα πεδία και οι μέθοδοι που αφορούν την δημιουργία και την εμφάνιση του παραθύρου.

`Texture2D defaultbg:` πεδίο, δυσδιάστατο πλέγμα εικονοστοιχείων της εικόνας παρασκηνίου του κεντρικού παραθύρου.

`Texture2D tablebg:` πεδίο, δυσδιάστατο πλέγμα εικονοστοιχείων της εικόνας παρασκηνίου του παραθύρου του τραπεζιού.

`SideBar sidebar:` πεδίο, το «κουτί» που θα φιλοξενήσει τα στοιχεία που χρησιμοποιούνται για την εισαγωγή των επίλογων πονταρίσματος από τον χρήστη.

`Button check:` πεδίο, μέσω αυτού του κουμπιού μπορεί να επιλέξει την κίνηση «ντούκου» κατά το πονταρισμα όπως αυτή ορίζεται στου κανόνες του παιχνιδιού.

Button call: πεδίο, μέσω αυτού του κουμπιού ο χρήσης μπορεί να επιλέξει την κίνηση «τα βλέπω» κατά το πονταρισμα όπως αυτή ορίζεται στου κανόνες του παιχνιδιού.

Button fold: πεδίο, με το πάτημα του πλήκτρου fold ο παίκτης αποχωρεί από την παρτίδα.

`Button raise:` πεδίο, ο παίκτης ανεβάζει το ποσό του πονταρίσματος.

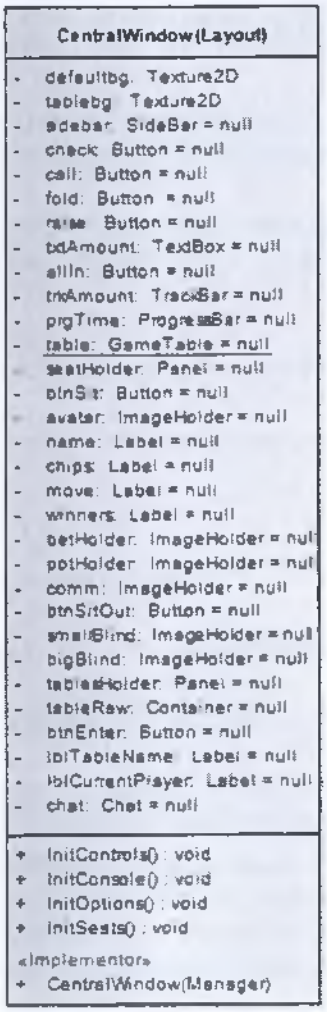
`TextBox textAmount:` πεδίο, το ποσό των εικονικών μαρκών που επιθυμεί να ποντάρει ο χρήστης.

`Button allIn:` πεδίο, μέσω αυτού του κουμπιού ο χρήσης μπορεί να επιλέξει την κίνηση «όλα μέσα» κατά το πονταρισμα όπως αυτή ορίζεται στου κανόνες του παιχνιδιού.

`TrackBar trkAmount`: πεδίο, μπάρα που επιτρέπει την επιλογή του πόσου πονταρίσματος.

`ProgressBar prgTime`: πεδίο, αντίστροφη μέτρηση του χρόνου που απομένει προκειμένου ο παίκτης να ολοκληρώσει το ποντάρισμα του.

`GameTable table`: πεδίο, η κλάση με τα δεδομένα του τραπεζιού.



Σχήμα 3.4-11: Κλάση CentralWindow

`Panel seatHolder`: πεδίο, ταμπλό που περικλείει όλα τα καθίσματα του τραπεζιού.

`Button btnSit`: πεδίο, χρησιμοποιείται για την αρχικοποίηση των καθισμάτων.

`ImageHolder avatar`: πεδίο, κλάση προσωρινής αποθήκευσης των δεδομένων.

`Label name`: πεδίο, κλάση προσωρινής αποθήκευσης των δεδομένων.

`Label chips`: πεδίο, κλάση προσωρινής αποθήκευσης των δεδομένων.

`Label move`: πεδίο, κλάση προσωρινής αποθήκευσης των δεδομένων.

`Label winners`: πεδίο, κλάση προσωρινής αποθήκευσης των δεδομένων.

`ImageHolder betHolder`: πεδίο, κλάση προσωρινής αποθήκευσης των δεδομένων.

`ImageHolder potHolder`: πεδίο, κλάση προσωρινής αποθήκευσης των δεδομένων.

`ImageHolder comm`: πεδίο, εδώ εμφανίζονται γραφικά τα κοινά φύλλα.

`Button btnSitOut`: πεδίο, πλήκτρο για την αποχώρηση του παίκτη από την θέση του.

`ImageHolder smallBlind`: πεδίο, το εικονίδιο του smallBlind.

`ImageHolder bigBlind`: πεδίο, το εικονίδιο του bigBlind.

`Panel tablesHolder`: πεδίο, το ταμπλό μέσα στο οποίο βρίσκονται οι πληροφορίες για τα ενεργά τραπέζια.

`Container tableRaw`: πεδίο, περιέχει τα στοιχεία για ένα ενεργό τραπέζι.

`Button btnEnter`: πεδίο, το πλήκτρο για την είσοδο του χρήστη στο συγκεκριμένο τραπέζι.

`Label lblTableName`: πεδίο, ετικέτα με το όνομα του συγκεκριμένου τραπεζιού.

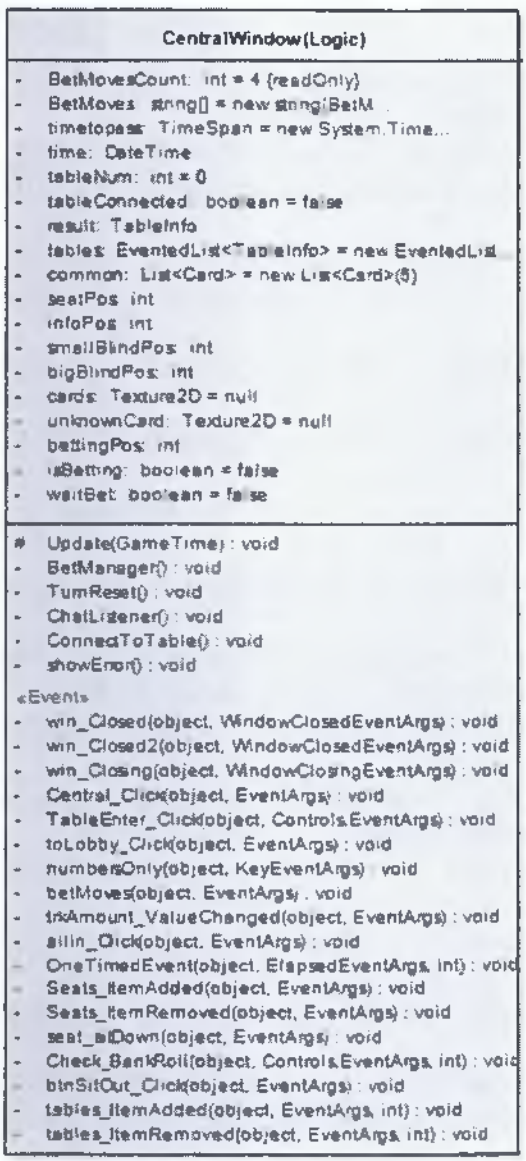
`Label lblCurrentPlayer`: πεδίο, οι ενεργοί παίκτες στο συγκεκριμένο τραπέζι.

`Chat chat`: πεδίο, η κλάση που διαχειρίζεται τα μηνύματα επικοινωνίας της εφαρμογής.

`InitControls`: μέθοδος, εδώ γίνεται η αρχικοποίηση των στοιχείων ελέγχου του παραθύρου.

`InitConsole`: μέθοδος, εδώ γίνεται η αρχικοποίηση της κονσόλας των μηνυμάτων επικοινωνίας.
`InitOptions`: μέθοδος, εδώ γίνεται η αρχικοποίηση των επιλογών πονταρίσματος.
`InitSeats`: μέθοδος, εδώ γίνεται η αρχικοποίηση των καθισμάτων του τραπεζιού.
`CentralWindow`: μέθοδος, χρησιμοποιείται για την δημιουργία και αρχικοποίηση των αντικειμένων της κλάσης.

`CentralWindow (Logic)`



Σχήμα 3.4-11: Κλάση CentralWindow

κλάση, σε αυτήν την κλάση υπάρχουν όλα τα αντικείμενα και οι μέθοδοι που χρησιμοποιούνται για την υλοποίηση της «λογικής» του προγράμματος.
`int BetMovesCount`: πεδίο, ο αριθμός των επολογών πονταρίσματος.
`string[] BetMoves`: πεδίο, οι επιλογές πονταρίσματος μέσα σε πίνακα συμβολοσειράς.
`TimeSpan timetopass`: πεδίο, χρησιμοποιείται για την δημιουργία χρονομέτρου πονταρίσματος.
`DateTime time`: πεδίο, χρησιμοποιείται για την δημιουργία χρονομέτρου πονταρίσματος.
`int tableNum`: πεδίο, ο αριθμός των ενεργών τραπεζιών.
`bool tableConnected`: πεδίο, δυαδική μεταβλητή αλήθειας, απαντά αν έχει γίνει σύνδεση σε τραπέζι.
`TableInfo result`: πεδίο, χρήση προσωρινής αποθήκευσης των πληροφοριών ενός τραπεζιού.
`EventedList<TablesInfo>` tables: πεδίο, η λίστα με τα ενεργά τραπέζια. Κάθε φορά που είτε προστίθεται είτε αφαιρείται ένα τραπέζι πυροδοτείται μια μέθοδος για την διαχείριση του γεγονότος.
`List<Card> common`: πεδίο, η λίστα με τα κοινά φύλλα.
`int seatPos`: πεδίο, προσωρινή μεταβλητή για την αποθήκευση θέσης.
`int infoPos`: πεδίο, προσωρινή μεταβλητή για την αποθήκευση θέσης.
`int smallBlindPos`: πεδίο, η θέση του

μικρού τυφλού ποντάρίσματος.
`int bigBlindPos`: πεδίο, η θέση του μεγάλου τυφλού ποντάρίσματος.
`Texture2D Cards`: πεδίο, δυσδιάστατο πλέγμα με τις εικόνες όλων των καρτών της τράπουλας.
`Texture2D unknownCard`: πεδίο, δυσδιάστατο πλέγμα με την εικόνα της πίσω όψης των καρτών.
`int bettingPos`: πεδίο, η θέση που ποντάρει την δεδομένη χρονική στιγμή.
`bool isBetting`: πεδίο, δυαδική μεταβλητή αλήθειας, απαντά αν πραγματοποιείται ποντάρισμα.
`bool waitBet`: πεδίο, δυαδική μεταβλητή αλήθειας, απαντά αν περιμένουμε για ποντάρισμα από παίκτη.
`Update`: μέθοδος, εδώ εκτελείται η «λογική» της εφαρμογής.
`BetManager`: μέθοδος, διαχειρίζεται το ποντάρισμα των παικτών.
`TurnReset`: μέθοδος, αρχικοποιεί τις μεταβλητές για την εκκίνηση νέας παρτίδας.
`ChatListener`: μέθοδος, είναι το «αυτί» που δέχεται τα εισερχόμενα μηνύματα επικοινωνίας.
`ConnectToTable`: μέθοδος, συνδέεται στον επιθυμητό εξυπηρετητή παιχνιδιού.
`showError`: μέθοδος, εμφανίζει σε πλαίσιο διαλόγου τα μηνύματα σφαλμάτων που προέρχονται από τον εξυπηρετητή.
`win_Closed`: μέθοδος, γεγονός που πυροδοτείται μετά τον τερματισμό μερικών παραθύρων.
`win_Closed2`: μέθοδος, γεγονός που πυροδοτείται μετά τον τερματισμό μερικών παραθύρων.
`win_Closing`: μέθοδος, γεγονός που πυροδοτείται κατά τη διάρκεια του τερματισμού μερικών παραθύρων.
`Central_Click`: μέθοδος, γεγονός που πυροδοτείται προκειμένου να τερματίσει το παράθυρο το οποίο πυροδοτεί το γεγονός.
`TableEnter_Click`: μέθοδος, γεγονός που πυροδοτείται από τα κουμπιά που χρησιμοποιούνται για την είσοδο σε τραπέζια παιχνιδιού.
`ToLobby_Click`: μέθοδος, γεγονός που πυροδοτείται από το κουμπί για την επιστροφή του χρήστη στο Ενρίκο παράθυρο-λόμπι
`numbersOnly`: μέθοδος, γεγονός που πυροδοτείται και επιβεβαιώνει την εισαγωγή μόνο αριθμητικών τιμών σε συγκεκριμένα πλαίσια κειμένου.
`betMoves`: μέθοδος, γεγονός που πυροδοτείται για τη διαχείριση της επιλογής ποντάρίσματος.
`trkAmount_ValueChanged`: μέθοδος, γεγονός που πυροδοτείται με την αλλαγή της τιμής της μπάρας ίχνους (TrackBar).
`allIn_Click`: μέθοδος, γεγονός που πυροδοτείται με την πίεση του κουμπιού "allIn".
`OneTimeEvent`: μέθοδος, γεγονός που πυροδοτείται με το πέρας του χρόνου ποντάρίσματος και το οποίο απορρίπτει τις κάρτες του συγκεκριμένου παίκτη από το χέρι του.
`Seats_ItemAdded`: μέθοδος, γεγονός που πυροδοτείται όταν προστίθεται ένα στοιχειό στη λίστα καθισμάτων.

`Seats_ItemRemoved`: μέθοδος, γεγονός που πυροδοτείται όταν αφαιρείται ένα στοιχείο από την λίστα καθισμάτων.
`Seats_sitDown`: μέθοδος, γεγονός που πυροδοτείται όταν ένας παίκτης κάθεται σε μια θέση.
`Check_BankRoll`: μέθοδος, γεγονός που πυροδοτείται και ελέγχει αν το ποσό των εικονικών χρημάτων που εισήγαγε ο παίκτης βρίσκεται στα όρια του τραπέζιου.
`btnSitOut_Click`: μέθοδος, γεγονός που πυροδοτείται για την αποχώρηση του συγκεκριμένου παίκτη από τη θέση του.
`tables_ItemAdded`: μέθοδος, γεγονός που πυροδοτείται όταν προστίθεται ένα στοιχειό στη λίστα tables.
`tables_ItemRemoved`: μέθοδος, γεγονός που πυροδοτείται όταν αφαιρείται ένα στοιχειό από τη λίστα tables.

-Κεφάλαιο 4°-

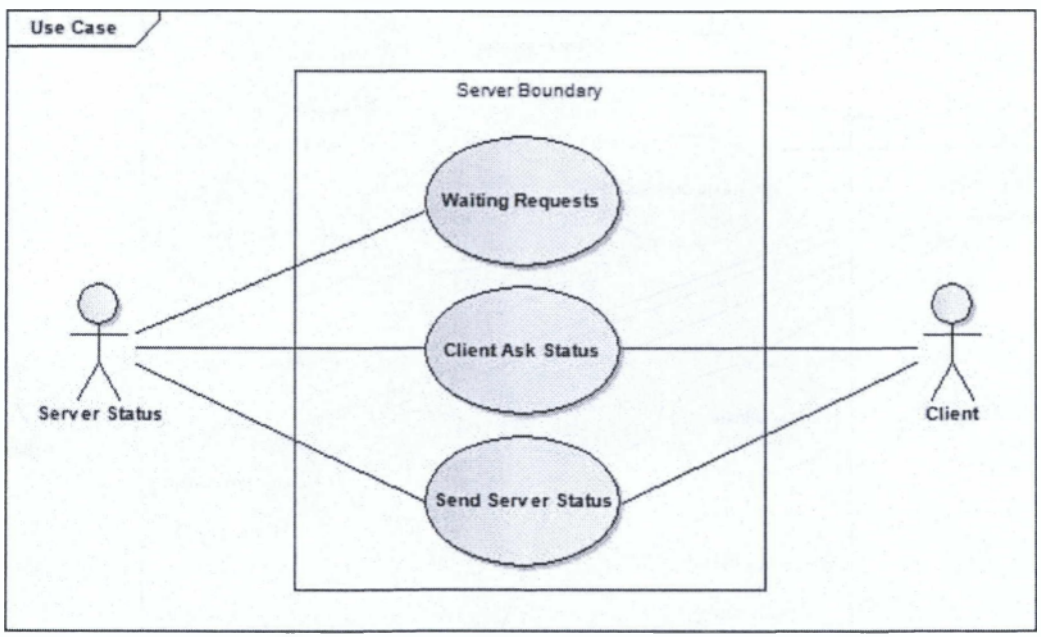
Δυναμικά Διαγράμματα

4. Δυναμικά Διαγράμματα

4.1 Διαγράμματα Περιπτώσεων

Τα διαγράμματα περιπτώσεων έχουν υλοποιηθεί για την ευκολότερη κατανόηση των βασικών λειτουργιών των εφαρμογών. Επίσης τα διαγράμματα υλοποιήθηκαν σε υψηλό επίπεδο αφαιρετικότητας. Για τον λόγο αυτό είναι πιθανόν να μην περιγράφουν όλες τις λειτουργιές που εκτελούν οι εφαρμογές όπως αυτές αναπτύχτηκαν στην φάση υλοποίησης. Ακόμα τα διαγράμματα δημιουργήθηκαν για τον σκοπό της απεικόνισης του τρόπου λειτουργίας τμημάτων του κώδικα και όχι την αυτούσια παρουσίαση του. Επομένως σε πολλές περιπτώσεις ενδέχεται η ιδέα ανάπτυξης του κώδικα να μην ταυτίζεται με αυτή των διαγραμμάτων.

4.1.1 Εξυπηρετητής κατάστασης

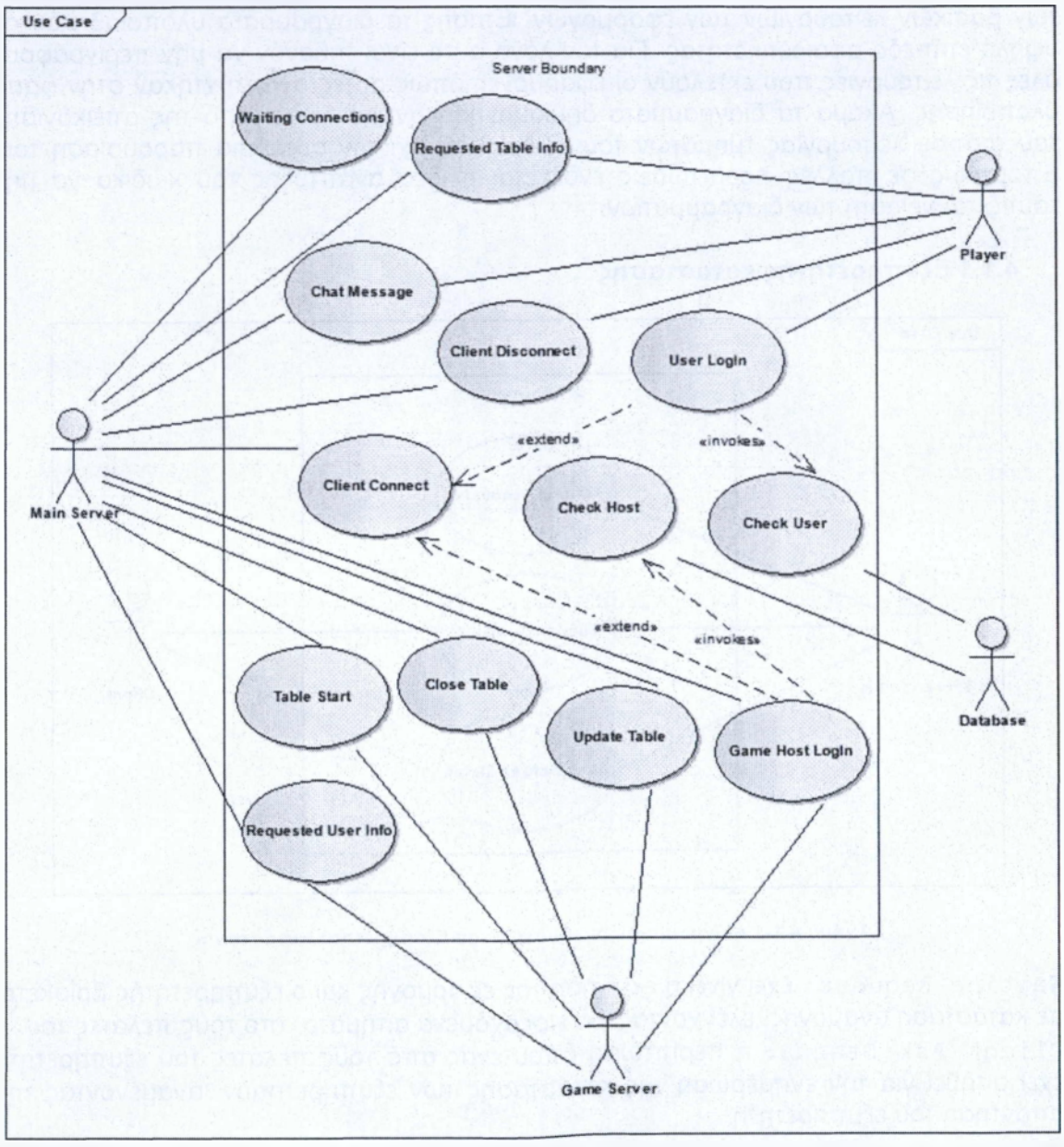


Σχήμα 4.1-1 Διάγραμμα περιπτώσεων του Εξυπηρετητή Κατάστασης

`Waiting Request`: έχει γίνει η εκκίνηση της εφαρμογής και ο εξυπηρετητής βρίσκεται σε κατάσταση αναμονής, ελέγχοντας για εισερχόμενα αιτήματα από τους πελάτες του.
`Client Ask Status`: η περίπτωση όπου ένας από τους πελάτες του εξυπηρετητή έχει αιτηθεί για την ενημέρωση της κατάστασης των εξυπηρετητών, αναμένοντας την απάντηση του εξυπηρετητή.
`Send Server Status`: η απάντηση του εξυπηρετητή σε έναν από τους πελάτες του, ο όποιος έχει αιτηθεί για την γνωστοποίηση της κατάστασης των εξυπηρετητών. Ο εξυπηρετητής συντάσσει ένα μήνυμα με τις πληροφορίες σχετικά με την κατάσταση των εξυπηρετητών και τον αριθμό της τελευταίας έκδοσης της εφαρμογής του πελάτη. Στην

παρούσα φάση οι πληροφορίες σχετικά με την τελευταία έκδοση δεν χρησιμοποιούνται από την εφαρμογή για την υλοποίηση κάποιας λειτουργίας.

4.1.2 Κεντρικός εξυπηρετητής



Σχήμα 4.1-2: Διάγραμμα περιπτώσεων του Κεντρικού Εξυπηρετητή

`Waiting Connection`: με την εκκίνηση της εφαρμογής ο κεντρικός εξυπηρετητής μετά την αρχικοποίηση μεταβαίνει σε κατάσταση αναμονής. Όμως καθ' όλη την διάρκεια

κατά την όποια βρίσκεται σε αναμονή ελέγχει για πιθανές εισερχόμενες συνδέσεις. Όταν μια νέα σύνδεση πραγματοποιείται, τα δεδομένα αυτής μεταφέρονται στον κατάλληλο χειριστή για την επεξεργασία τους.

`Client Connect`: ο πελάτης αιτείται προς τον κεντρικό εξυπηρετητή για την σύνδεση του σε αυτόν. Η διαδικασία σύνδεσης περιλαμβάνει και προϋποθέτει την διαδικασία πιστοποίησης του πελάτη στον κεντρικό εξυπηρετητή.

`User Login`: η περίπτωση χρήσης η όποια «εκτελεί» τη διαδικασία της συλλογής των στοιχείων που απαιτούνται για την πιστοποίηση του χρήστη στον κεντρικό εξυπηρετητή.

`Game Host Login`: η περίπτωση χρήσης η όποια «εκτελεί» την διαδικασία της συλλογής των δεδομένων και του ελέγχου του εξυπηρετητή που επιθυμεί να φιλοξενήσει ένα τραπέζι.

`Check User`: η συγκεκριμένη περίπτωση χρήσης περιγράφει την διαδικασία ελέγχου των στοιχείων του χρήστη με αυτά της βάσης δεδομένων. Μετά την επιτυχή πιστοποίηση του χρήστη, επιτρέπεται σε αυτόν η πρόσβαση του σε δεδομένα της εφαρμογής ανάλογα με το επίπεδο πρόσβασης του.

`Check Host`: η συγκεκριμένη περίπτωση χρήσης περιγράφει την διαδικασία ελέγχου των στοιχείων του εξυπηρετητή που επιθυμεί να φιλοξενήσει κάποιο τραπέζι, με αυτά της βάσης δεδομένων.

`Requested Table Info`: ο χρήστης ζητεί από τον κεντρικό εξυπηρετητή τις πληροφορίες με τα ενεργά τραπέζια. Ο εξυπηρετητής στέλνει σε αυτόν τις πληροφορίες. Ταυτόχρονα εγγράφει τον χρήστη σε μια λίστα για τη μετέπειτα ενημέρωση του σχετικά με τις μεταβολές των πληροφοριών των τραπεζιών.

`Chat Message`: εδώ βρίσκεται η περίπτωση όπου ένας χρήστης έχει πληκτρολογήσει ένα κείμενο στην γραφική κονσόλα επικοινωνίας. Το μήνυμα του χρήστη μεταβιβάζεται στον κεντρικό εξυπηρετητή και αυτός το μεταβιβάζει σε όλους τους χρήστες της εφαρμογής.

`Client Disconnect`: όταν ένας πελάτης τερματίσει την εφαρμογή ή προκύψει κάποιο σφάλμα το όποιο δεν επιτρέπει την επικοινωνία με τον κεντρικό εξυπηρετητή, τότε ο κεντρικός εξυπηρετητής εκκινεί τις διαδικασίες αποσύνδεσης του από το σύστημα.

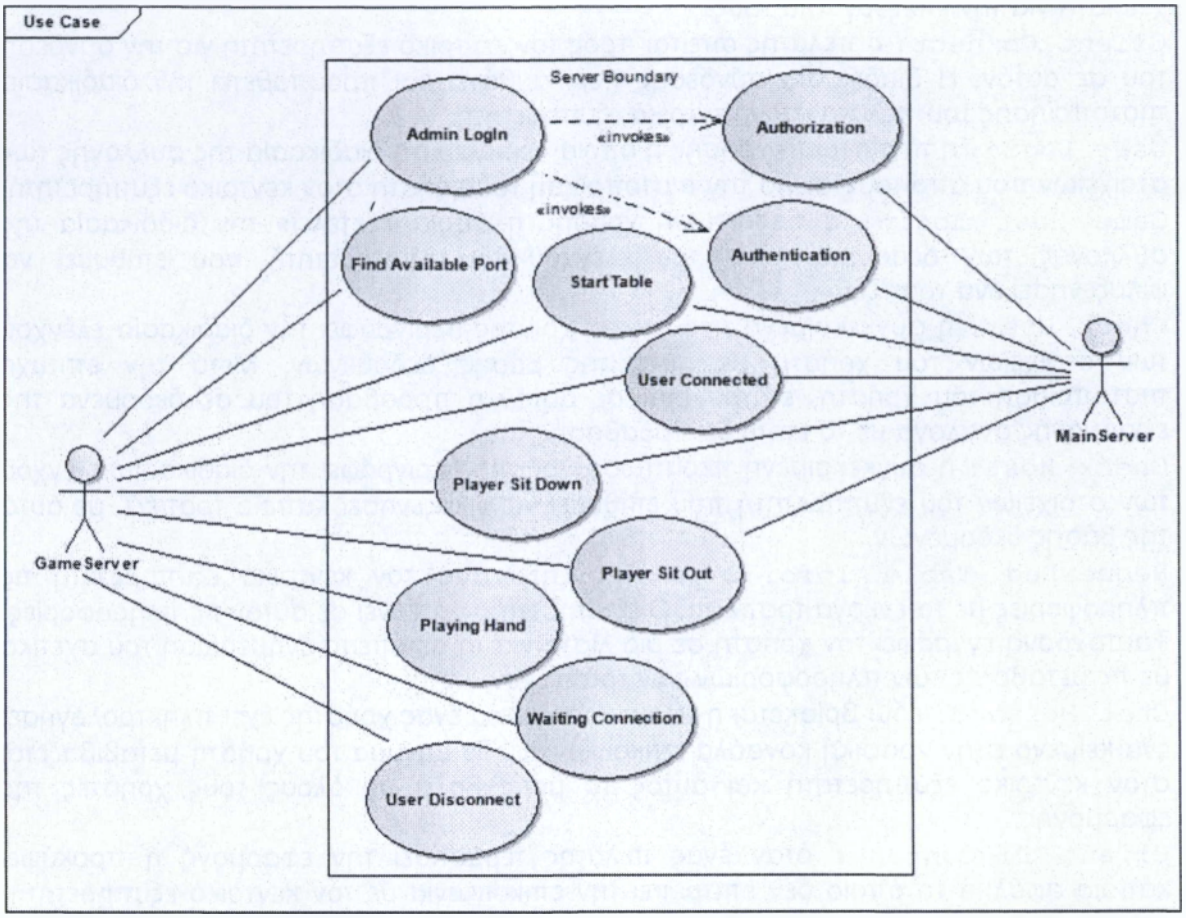
`Requested User Info`: οι εξυπηρετητές που φιλοξενούν τα παιχνίδια αιτούνται στον κεντρικό εξυπηρετητή προκειμένου να λάβουν τα στοιχεία των χρηστών που επιχειρούν να συνδεθούν σε αυτούς.

`Table Start`: η διαδικασία εκκίνησης ενός τραπεζιού. Οι πληροφορίες του εξυπηρετητή παιχνιδιού μεταφέρονται στον κεντρικό εξυπηρετητή για μετέπειτα χρήση.

`Update Table`: όταν συμβαίνει μια μεταβολή σε ένα από τα τραπέζια, ο εξυπηρετητής του συγκεκριμένου παιχνιδιού, στέλνει τις πληροφορίες στον κεντρικό εξυπηρετητή.

`Close Table`: όταν κλείνει η εφαρμογή του εξυπηρετητή παιχνιδιού ή χάνεται η επικοινωνία του με τον κεντρικό εξυπηρετητή εκκινείται η διαδικασία τερματισμού του τραπεζιού που φιλοξενείται από αυτόν.

4.1.3 Εξυπηρετητής παιχνιδιού



Σχήμα 4.1-3: Διάγραμμα περιπτώσεων του Εξυπηρετητή Κατάστασης

`Admin Login`: κατά το «άνοιγμα» της εφαρμογής του εξυπηρετητή παιχνιδιού εκκινείται αρχικά η διαδικασία πιστοποίησης των στοιχείων του εξυπηρετητή παιχνιδιού και έπειτα η διαδικασία ελέγχου της εξουσιοδότησης του εξυπηρετητή παιχνιδιού.
`Authentication`: τα στοιχεία (ψευδώνυμο, συνθηματικό) που στέλνονται στον κεντρικό εξυπηρετητή από τον εξυπηρετητή παιχνιδιού ελέγχονται για την ορθότητα τους. Έπειτα πραγματοποιείται η σύνδεση του εξυπηρετητή παιχνιδιού στον κεντρικό εξυπηρετητή.
`Authorization`: η πρόσβαση που αποκτάται από τον εξυπηρετητή παιχνιδιού σε ορισμένες λειτουργίες του κεντρικού εξυπηρετητή. Πρώτα ελέγχεται αν ο συγκεκριμένος εξυπηρετητής παιχνιδιού έχει τα απαραίτητα δικαιώματα. Τα κριτήρια ελέγχου και απόκτησης πρόσβασης είναι καθαρά ποιοτικά.
`Find Available Port`: γίνεται αναζήτηση για μια ελεύθερη πόρτα στο σύστημα του εξυπηρετητή παιχνιδιού, προκειμένου να μπορεί να γίνει η επικοινωνία του με τους

πελάτες του. Η πόρτα καθώς και η ίδια η εφαρμογή του κεντρικού εξυπηρετητή πρέπει να μην εμποδίζεται από τοίχους προστασίας και προγράμματα προστασίας από ιούς.

`Start Table`: μετά την απόκτησης δικαιώματος φιλοξενίας ενός παιχνιδιού γίνεται η εκκίνηση του τραπεζιού.

`User Connected`: η περίπτωση και οι ενέργειες που θα πραγματοποιηθούν κατά την σύνδεση ενός παίκτη στον συγκεκριμένο εξυπηρετητή παιχνιδιού.

`Player SitDown`: όλες οι ενέργειες που γίνονται από τον παίκτη που αιτείται την απόκτηση μιας θέσης στον συγκεκριμένο εξυπηρετητή παιχνιδιού. Μετά τους απαραιτήτους ελέγχους, αν δεν υπάρξει σφάλμα ο παίκτης κάθεται στη δεδομένη θέση.

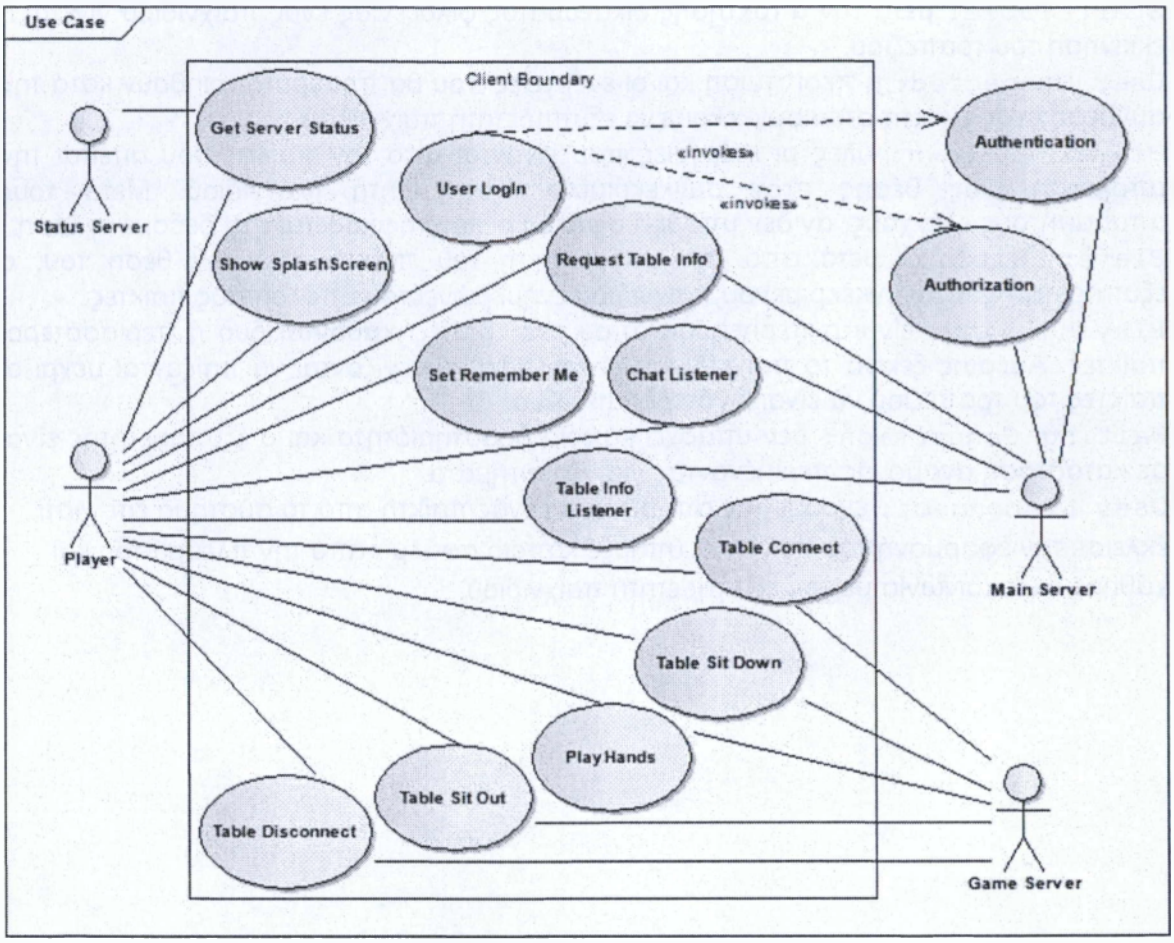
`Player SitOut`: μετά από την αποχώρηση του παίκτη από την θέση του, ο εξυπηρετητής του συγκεκριμένου παιχνιδιού ενημερώνει του υπόλοιπους παίκτες.

`Playing Hand`: είναι η περίπτωση όπου στο τραπέζι κάθονται δυο ή περισσότεροι παίκτες. Αμέσως ξεκινά το παιχνίδι και οι παρτίδες συνεχίζονται να παίζονται μέχρι οι παίκτες του τραπεζιού να είναι λιγότεροι από δυο.

`Waiting Connection`: δεν υπάρχει κάποια δραστηριότητα και ο εξυπηρετητής είναι σε κατάσταση αναμονής περιμένοντας για νέα αιτήματα.

`User Disconnect`: έχουμε την αποσύνδεση ενός παίκτη από το σύστημα είτε γιατί έκλεισε την εφαρμογή του πελάτη ή υπήρξε κάποιο σφάλμα από την πλευρά του και χάθηκε η επικοινωνία με τον εξυπηρετητή παιχνιδιού.

4.1.4 Εφαρμογή πελάτη



Σχήμα 4.1-4: Διάγραμμα περιπτώσεων της εφαρμογής πελάτη

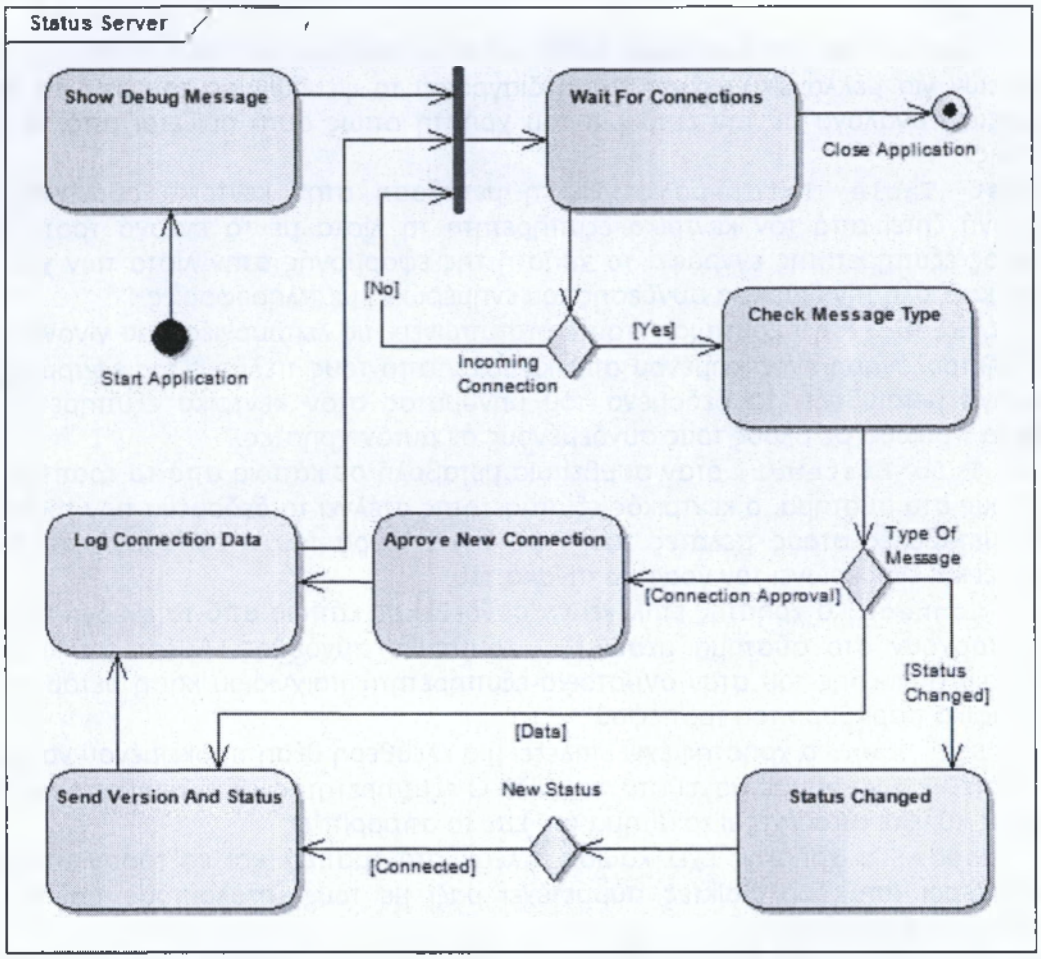
`Show SplashScreen`: κατά το άνοιγμα της εφαρμογής εμφανίζεται μια οθόνη μέσα στην όποια εμφανίζεται ένα ενημερωτικό κείμενο σχετικά με τον σκοπό μέσα στα πλαίσια του οποίου έγινε η ανάπτυξη της εφαρμογής.
`Get Server Status`: μετά το πέρας της οθόνης υποδοχής η εφαρμογή επικοινωνεί με τον εξυπηρετητή κατάστασης προκειμένου να γίνει ενημέρωση σχετικά με τις πληροφορίες των εξυπηρετητών στην δεδομένη χρονική στιγμή.
`User Login`: πλέον έχει εμφανιστεί η οθόνη σύνδεσης και εφ' όσον οι εξυπηρετητές είναι διαθέσιμοι, τότε ο χρήστης μπορεί να επιχειρήσει να συνδεθεί στο σύστημα του κεντρικού εξυπηρετητή.
`Authentication`: εάν ένας χρήστης προσπαθήσει να συνδεθεί ο κεντρικός εξυπηρετητής επεξεργάζεται τα δεδομένα που δέχεται και αν ταιριάζουν με αυτά που βρίσκονται στην βάση με την όποια επικοινωνεί τότε η σύνδεση του θεωρείται επιτυχής και ο χρήστης μεταβαίνει στην κεντρική οθόνη-λόμπι. Σε αντίθετη περίπτωση ο κεντρικός εξυπηρετητής αρνείται την πρόσβαση στα δεδομένα της εφαρμογής.

`Authorization:` με την επιτυχή σύνδεση ενός χρήστη γίνεται έλεγχος για την εξουσιοδότηση και τα δικαιώματα πρόσβασης αυτού στις λειτουργίες του κεντρικού εξυπηρετητή.
`Set Remember Me:` η εφαρμογή αποθηκεύει το ψευδώνυμο του χρήστη σε αρχείο ρυθμίσεων για μελλοντική χρήση του ή διαγράφει το ψευδώνυμο του από το αρχείο ρυθμίσεων, ανάλογα με την επιθυμία του χρήστη όπως αυτή ορίζεται από το πεδίο επιλογής.
`Request Table Info:` μόλις γίνει η μετάβαση στην κεντρική οθόνη-λόμπι η εφαρμογή ζητεί από τον κεντρικό εξυπηρετητή τη λίστα με τα ενεργά τραπέζια. Ο κεντρικός εξυπηρετητής εγγράφει το χρήστη της εφαρμογής στην λίστα των χρηστών για την καθ' όλη την διάρκεια σύνδεσης του ενημέρωση με πληροφορίες.
`Chat Listener:` η περίπτωση που «αποτυπώνει» τις λειτουργιές που γίνονται μετά την πληκτρολόγηση ενός κειμένου από κάποιον από τους πελάτες της εφαρμογής. Η εφαρμογή μεταβιβάζει τα δεδομένα του μηνύματος στον κεντρικό εξυπηρετητή και αυτός τα προωθεί σε όλους τους συνδεμένους σε αυτόν χρήστες.
`Table Info Listener:` όταν συμβεί μια μεταβολή σε κάποιο από τα τραπέζια που υπάρχουν στο σύστημα, ο κεντρικός εξυπηρετητής στέλνει τα δεδομένα των τραπεζιών με τις μεταβολές στους πελάτες του. Από την πλευρά του η εφαρμογή πελάτη τις λαμβάνει και ενημερώνει τον γραφικό πίνακα του.
`Table Connect:` ο χρήστης επιλεγεί να συνδεθεί σε κάποιο από τα ενεργά τραπέζια που υπάρχουν στο σύστημα μέσω των κουμπιών σύνδεσης. Αμέσως εκκινείται η διαδικασία σύνδεσης του στον αντίστοιχο εξυπηρετητή παιχνιδιού και η μετάβαση του στο γραφικό παράθυρο του τραπεζιού.
`Table Sit Down:` ο χρήστης έχει επιλέξει μια ελεύθερη θέση προκειμένου να καθίσει στο τραπέζι και να συμμετάσχει στο παιχνίδι. Ο εξυπηρετητής κάνει τους απαραίτητους ελέγχους και είτε αποδέχεται το αίτημα του είτε το απορρίπτει.
`Play Hands:` ο χρήστης έχει καθίσει πλέον στο τραπέζι και εφ' όσον υπάρχουν περισσότεροι από δυο παίκτες συμμετέχει μαζί με τους υπόλοιπους παίκτες στις παρτίδες του παιχνιδιού.
`Table Sit Out:` ένας παίκτης αποχωρεί από το τραπέζι, αμέσως γίνεται εκκαθάριση των στοιχείων που αφορούσαν στον συγκεκριμένο παίκτη από το τραπέζι.
`Table Disconnect:` έχουμε την αποσύνδεση του χρήστη από το τραπέζι και είτε την επιστροφή του στην κεντρική οθόνη-λόμπι είτε τον τερματισμό της εφαρμογής πελάτη.

4.2 Διάγραμμα Δραστηριοτήτων

Τα διαγράμματα δραστηριοτήτων είναι σχεδιαγράμματα τα όποια απεικονίζουν βήμα-βήμα τη λειτουργία των περιεχόμενων ενός συστήματος. Στο σύνολο τους τα διαγράμματα παρουσιάζουν την συνολική ροή του ελέγχου μέσα στο περιγραφόμενο σύστημα. Στην συγκεκριμένη εργασία θα χρησιμοποιήσουμε τα διαγράμματα δραστηριοτήτων για την ανάλυση της πορείας του ελέγχου των βασικότερων λειτουργιών και των περιεχόμενων του συστήματος χωρίς παρ' όλα αυτά να περιγράφεται το σύνολο τον ροών ελέγχου του συστήματος.

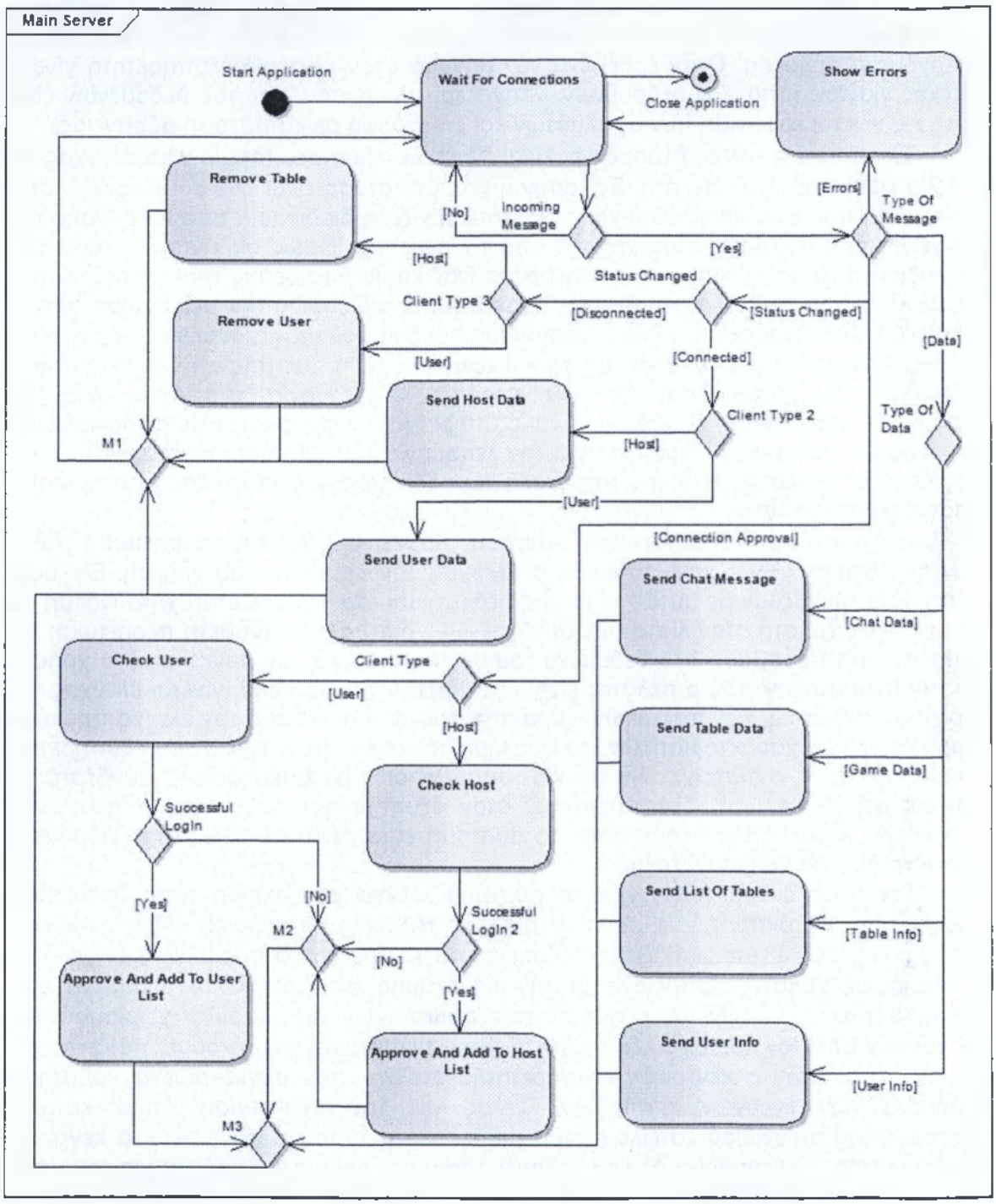
4.2.1 Εξυπηρετητής κατάστασης



Σχήμα 4.2-1: Διάγραμμα δραστηριοτήτων του Εξυπηρετητή Κατάστασης

Στο παραπάνω διάγραμμα η ροή ελέγχου ξεκινά με το «άνοιγμα» της εφαρμογής και την εμφάνιση ενός μηνύματος απασφαλμάτωσης που αφορά στην αρχικοποίηση της εφαρμογής. Έπειτα το σύστημα βρίσκεται σε κατάσταση αδράνειας περιμένοντας για εισερχόμενες συνδέσεις και αιτήματα. Όταν μια εισερχόμενη σύνδεση εντοπιστεί γίνεται έλεγχος για τον τύπο του μηνύματος. Αν ο τύπος του μηνύματος είναι δεδομένα τύπου `Connection Approval` τότε γίνεται αποδοχή της σύνδεσης, καταγραφή των στοιχείων του πελάτη που επιχείρησε να συνδεθεί και μετάβαση της ροής σε κατάσταση αναμονής. Αν ο τύπος δεδομένων είναι τύπου `Data` ο εξυπηρετητής κατάστασης στέλνει στον πελάτη που αιτήθηκε τα δεδομένα με τα στοιχεία του αριθμού της έκδοσης και της κατάστασης των εξυπηρετητών. Τέλος αν ο τύπος δεδομένων είναι `Status Changed`, αν η νέα κατάσταση του πελάτη ο όποιος «πυροδότησε» την αλλαγή κατάστασης είναι `Connected` γίνεται αποστολή των δεδομένων με τα στοιχεία του αριθμού έκδοσης και της κατάστασης των εξυπηρετητών.

4.2.2 Κεντρικός εξυπηρετητής



Σχήμα 4.2-2: Διάγραμμα δραστηριοτήτων του Κεντρικού Εξυπηρετητή

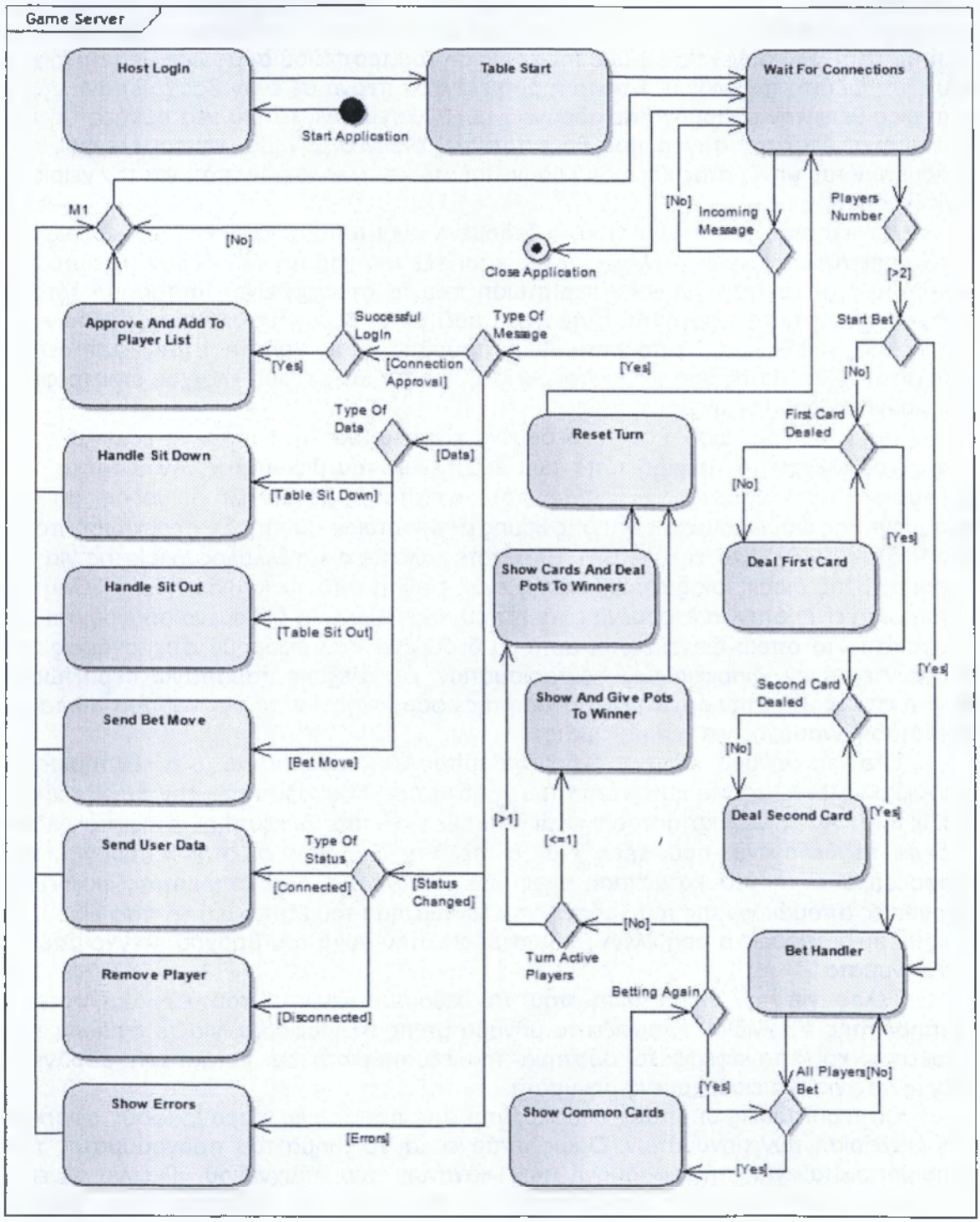
Συμφώνα με το παραπάνω διάγραμμα, η λειτουργία του κεντρικού εξυπηρετητή γίνεται με την εκκίνηση και αρχικοποίηση του προγράμματος. Μετά ο κεντρικός εξυπηρετητής περνά σε κατάσταση αναμονής μέχρι τη στιγμή που θα εντοπιστεί μια εισερχόμενη σύνδεση. Όταν έρθει ένα νέο μήνυμα στον κεντρικό εξυπηρετητή γίνεται έλεγχος για τον τύπο των δεδομένων. Στην περίπτωση που ο τύπος δεδομένων είναι `Errors`, γίνεται εμφάνιση των σφαλμάτων και επιστροφή σε κατάσταση αδράνειας.

Έπειτα αν ο τύπος δεδομένων είναι `Status Change`, τότε γίνεται έλεγχος για την νέα κατάσταση του πελάτη. Εφ' όσον η νέα κατάσταση είναι `Disconnected`, στην περίπτωση που ο πελάτης είναι χρήστης, τότε αρχίζει η διαδικασία απομάκρυνσης των στοιχείων που αφορούν στον χρήστη από το σύστημα. Όμως αν είναι εξυπηρετητής παιχνιδιού, τότε το σύστημα προχωρεί στην διαδικασία αφαίρεσης του τραπεζιού που φιλοξενείται από των συγκεκριμένο εξυπηρετητή παιχνιδιού. Για την περίπτωση όπου η νέα κατάσταση είναι `Connected`, ακολουθούνται δυο μονοπάτια ανάλογα με τύπο του πελάτη. Αν ο πελάτης είναι χρήστης, τότε ο κεντρικός εξυπηρετητής στέλνει σε αυτόν τα δεδομένα που απαιτούνται για την περαιτέρω λειτουργία της εφαρμογής. Σε διαφορετική περίπτωση, δηλαδή αν είναι εξυπηρετητής παιχνιδιού, τότε αποστέλλονται τα δεδομένα που είναι απαραίτητα για την λειτουργία του εξυπηρετητή παιχνιδιού. Και στις δυο περιπτώσεις μετά την αποστολή των δεδομένων ο κεντρικός εξυπηρετητής επιστρέφει σε κατάσταση αδράνειας.

Για τον τύπο δεδομένων `Connection Approval` και στην περίπτωση που ο πελάτης είναι απλά χρήστης, τότε γίνεται έλεγχος των στοιχείων του χρήστη. Εφ' όσον τα στοιχεία ταιριάζουν με αυτά της βάσης δεδομένων τότε έχουμε επιτυχή σύνδεση και εγγραφή του χρήστη στην λίστα των συνδεμένων χρηστών. Σε αντίθετη περίπτωση δεν επιτρέπεται η πρόσβαση στα δεδομένα του συστήματος για τον συγκεκριμένο χρήστη. Για την περίπτωση όπου ο πελάτης είναι εξυπηρετητής παιχνιδιού, γίνεται έλεγχος των στοιχείων καθώς και των ποιοτικών χαρακτηριστικών του. Αν από τον έλεγχο προκύψει η ορθότητα των χαρακτηριστικών, το σύστημα τοποθετεί τα στοιχεία του εξυπηρετητή παιχνιδιού στην αντίστοιχη λίστα, και αποδέχεται την εισερχόμενη σύνδεση. Σε διαφορετική περίπτωση δεν παρέχεται στον εξυπηρετητή το δικαίωμα φιλοξενίας παιχνιδιών. Σε όλες της περιπτώσεις το σύστημα επιστρέφει σε κατάσταση αδράνειας περιμένοντας για νέες συνδέσεις.

Για τύπο δεδομένων `Data` το σύστημα ελέγχει τον ακριβή τύπο δεδομένων προκειμένου να διαπιστώσει σε ποια από τις τέσσερις κατηγορίες ανήκει. Αν είναι τύπου `Chat Data` τότε διαβάζει τα δεδομένα του χρήστη και τα προωθεί σε όλους τους συνδεμένους χρήστες. Όταν έχουμε την περίπτωση στην όποια τα δεδομένα είναι τύπου `Game Data` γίνεται εκκίνηση των αντίστοιχων διαδικασιών χειρισμού των δεδομένων από τον κεντρικό εξυπηρετητή. Στην περίπτωση που έχουμε αίτηση τύπου `Table Info` τότε ο κεντρικός εξυπηρετητής στέλνει στον συγκεκριμένο χρήστη τα δεδομένα των ενεργών τραπεζιών. Τέλος για την περίπτωση όπου κάποιος εξυπηρετητής παιχνιδιού έστειλε αίτημα με δεδομένα τύπου `User Info` ο κεντρικός εξυπηρετητής συγκεντρώνει όλα τα δεδομένα που αφορούν στον χρήστη για τον όποιο έγινε το αίτημα και τα στέλνει στον εξυπηρετητή παιχνιδιού αυτά τα δεδομένα. Σε όλες τις προαναφερθέντες περιπτώσεις μετά τη διενέργεια των απαραίτητων διαδικασιών, η ροή ελέγχου του κεντρικού εξυπηρετητή θέτει αυτόν σε κατάσταση αναμονής.

4.2.3 Εξυπηρετητής παιχνιδιού



Σχήμα 4.2-3: Διάγραμμα δραστηριοτήτων του Εξυπηρετητή Παιχνιδιού

Στο παραπάνω διάγραμμα απεικονίζεται η ροή ελέγχου του εξυπηρετητή παιχνιδιού. Η ροή ελέγχου ξεκινά με την εκκίνηση της εφαρμογής και την εισαγωγή των στοιχείων σύνδεσης και την σύνδεση του εξυπηρετητή παιχνιδιού στον κεντρικό εξυπηρετητή. Στη συνεχεία έχουμε την εκκίνηση του τραπεζιού συμφώνα με τα στοιχεία του εξυπηρετητή παιχνιδιού. Έπειτα η ροή ελέγχου περνά σε έναν βρόχο επανάληψης ο όποιος θέτει την εφαρμογή σε αδράνεια μέχρι να εμφανιστεί μια νέα σύνδεση ή ένα νέο αίτημα. Από την στιγμή που θα εντοπιστεί ένα νέο μήνυμα, γίνεται έλεγχος των δεδομένων του μηνύματος και η ροή οδηγείται στο κατάλληλο μονοπάτι για τον χειρισμό των δεδομένων.

Αρχικά στην περίπτωση που τα δεδομένα είναι τύπου `Connection Approval` ο εξυπηρετητής παιχνιδιού ελέγχει αν τα στοιχεία του χρήστη ταιριάζουν με αυτά του κεντρικού εξυπηρετητή και στην περίπτωση που τα στοιχεία είναι ταυτόσημα τότε τα στοιχεία του χρήστη εισάγονται στην λίστα μαζί με αυτά των υπολοίπων παικτών. Σε διαφορετική περίπτωση, απορρίπτεται η πράσβαση του χρήστη στον εξυπηρετητή παιχνιδιού. Και για τις δυο προαναφερθείσες περιπτώσεις η ροή ελέγχου επιστρέφεται στο βρόχο ελέγχου νέων μηνυμάτων.

Για την περίπτωση που τα δεδομένα είναι τύπου `Data` τότε ο εξυπηρετητής παιχνιδιού ελέγχει τον ακριβή τύπο των δεδομένων του μηνύματος. Αν ο τύπος των δεδομένων είναι `Table Sit Down` τότε καλείται η κατάλληλη μέθοδος για την εφαρμογή της διαδικασίας απόκτησης θέσης από κάποιον παίκτη. Σε περίπτωση που ο τύπος δεδομένων είναι `Table Sit Out` τότε καλείται ο κατάλληλος χειρίστης για την εφαρμογή της διαδικασίας αποχώρησης ενός παίκτη από το κάθισμα του. Τελευταία περίπτωση είναι όταν τα δεδομένα είναι τύπου `Bet Move`, τα δεδομένα οδηγούνται στο «μονοπάτι» το όποιο διαχειρίζεται αυτά τα δεδομένα που αφορούν στις κινήσεις των παικτών κατά την διάρκεια των πονταρισμάτων. Για όλες τις παραπάνω περιπτώσεις γίνεται επιστροφή στην αρχική κατάσταση της εφαρμογής, και το πρόγραμμα τίθεται σε κατάσταση αναμονής για νέα μηνύματα.

Όταν το μήνυμα περιέχει δεδομένα τύπου `Status Changed` ο εξυπηρετητής παιχνιδιού ελέγχει τη νέα κατάσταση του χρήστη που «πυροδότησε» την συγκεκριμένη διαδικασία. Αν η νέα κατάσταση είναι `Connected` τότε ο εξυπηρετητής παιχνιδιού στέλνει τα δεδομένα που χρειάζεται ο πελάτης για την σωστή λειτουργία του. Διαφορετικά αν η νέα κατάσταση είναι `Disconnected` ο εξυπηρετητής εκκινεί τις διεργασίες απομάκρυνσης του χρήστη από το σύστημα του εξυπηρετητή παιχνιδιού. Σε όλες τις περιπτώσεις η ροή ελέγχου επιστρέφει στην αρχή του βρόχου ελέγχοντας για νέα μηνύματα.

Τέλος για την περίπτωση που τα δεδομένα είναι τύπου `Errors` τότε, ο εξυπηρετητής παιχνιδιού εμφανίζει το μήνυμα με τις πληροφορίες για το σφάλμα που προέκυψε και επαναφέρει το σύστημα του εξυπηρετητή σε κατάσταση αδράνειας ελέγχοντας για νέα εισερχόμενα μηνύματα.

Οι περιπτώσεις οι οποίες αναφέρονται στις παραπάνω παραγράφους αφορούν στη διαχείριση των μηνυμάτων. Όμως υπάρχει και το τμήμα του προγράμματος που χρησιμοποιείται για την εφαρμογή των κανόνων του παιχνιδιού. Για να γίνει η «εκτέλεση» των κανόνων του παιχνιδιού απαιτείται να κάθονται στο τραπέζι τουλάχιστον δυο παίκτες.

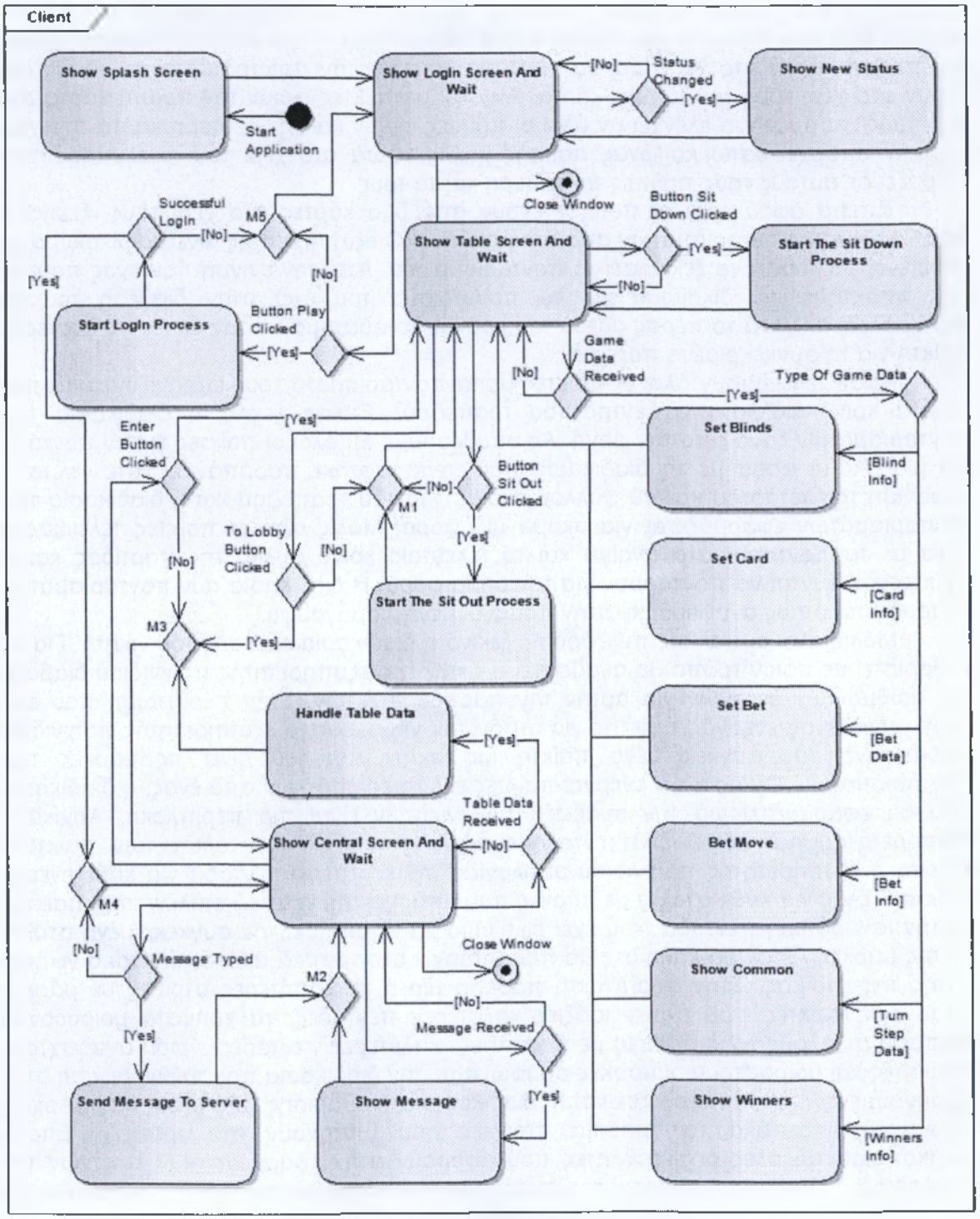
Από την στιγμή που υπάρχουν δυο ή περισσότεροι παίκτες στο τραπέζι, τότε το σύστημα κάνει ελέγχους για να διαπιστώσει αν πρέπει να ξεκινήσουν τα πονταρίσματα. Έπειτα αν δεν πρέπει να ξεκινήσουν τα πονταρίσματα γίνεται έλεγχος αν ο κάθε παίκτης έχει στο χέρι του την πρώτη κάρτα. Αν υπάρχει έστω και ένας παίκτης χωρίς την πρώτη κάρτα στο χέρι του, το σύστημα μοιράζει την πρώτη κάρτα σε όσους δεν έχουν στο χέρι τους την πρώτη κάρτα. Αν όλοι οι παίκτες έχουν την πρώτη κάρτα στο χέρι τους, το σύστημα ελέγχει αν όλοι οι παίκτες έχουν και την δεύτερη κάρτα στο χέρι του. Αν υπάρχει έστω και ένας παίκτης χωρίς κάρτα στο χέρι του, ο εξυπηρετητής μοιράζει σε αυτούς τους παίκτες τη δεύτερη κάρτα τους.

Έπειτα αφού όλοι οι παίκτες έχουν από δυο κάρτες στο χέρι τους, ξεκινά η διαδικασία των πονταρισμάτων από τους παίκτες. Ο εξυπηρετητής δίνει το δικαίωμα σε κάθε ενεργό παίκτη να ξεκινήσει το ποντάρισμα του. Από την στιγμή που ένας παίκτης έχει αποκτήσει το δικαίωμα για το ποντάρισμα του έχει στην διάθεση του 60 δευτερόλεπτα. Μετά το πέρας αυτού του χρόνου το σύστημα θα τον θέσει ως ανενεργό παίκτη για τη συγκεκριμένη παρτίδα.

Αφού τελειώσουν όλοι οι παίκτες με τα πονταρίσματα τους εμφανίζονται τα τρία πρώτα κοινά φύλλα στο κέντρο του τραπεζιού. Έπειτα ξεκινά η διαδικασία των πονταρισμάτων ξανά από την αρχή. Ας υποθέσουμε ότι όλοι οι παίκτες έχουν ποντάρει για μία ακόμα φορά με τη διαδικασία που περιγράφτηκε παραπάνω. Τότε γίνεται η εμφάνιση του τέταρτου κοινού φύλλου στο κέντρο του τραπεζιού και η διαδικασία των πονταρισμάτων εφαρμόζεται για ακόμα μια φορά. Μόλις όλοι οι παίκτες τελειώσουν ξανά με τα πονταρίσματα ανοίγει και το τελευταίο κοινό φύλλο της παρτίδας και οι παίκτες καλούνται να ποντάρουν για τελευταία φορά. Η διαδικασία των πονταρισμάτων εφαρμόζεται όπως αναφέρθηκε στην προηγούμενη παράγραφο.

Μόλις όλοι οι παίκτες ποντάρουν, ξεκινά η διαδικασία εύρεσης του νικητή. Για να καθοριστεί με ποιον τρόπο θα αναδειχτεί ο νικητής, ο εξυπηρετητής παιχνιδιού διαβάζει τον αριθμό των ενεργών για αυτήν την παρτίδα παικτών. Στην περίπτωση που έχει μείνει μόνο ένας ενεργός παίκτης για αυτόν τον γύρο, τότε ο εξυπηρετητής παιχνιδιού ανακοινώνει τον συγκεκριμένο παίκτη ως νικητή και του δίνει τις μάρκες των πονταρισμάτων. Όμως αν οι ενεργοί παίκτες είναι περισσότεροι από ένας, η διαδικασία που θα εφαρμοστεί για την ανάδειξη των νικητών είναι πιο περιπλοκή. Αρχικά ο εξυπηρετητής παιχνιδιού δίνει εντολή εμφάνισης των φύλλων όλων των παικτών. Έπειτα ο εξυπηρετητής παιχνιδιού υπολογίζει την καλύτερη πεντάδα για κάθε ενεργό παίκτη. Τέλος για κάθε στοίβα με μάρκες που υπάρχει στο τραπέζι επιλέγεται ο παίκτης με την ισχυρότερη πεντάδα που έχει δικαίωμα να αποκτήσει την συγκεκριμένη στοίβα με της μάρκες. Έτσι για κάθε στοίβα που υπάρχει στο τραπέζι ανακοινώνεται ο νικητής αυτής της στοίβας. Στην περίπτωση που για μια ή περισσότερες στοίβες με μάρκες υπάρχουν παίκτες που έχουν ισάξιες καλύτερες πεντάδες, τα χρήματα μοιράζονται ισόποσα σε όλους τους παίκτες με τις ισάξιες καλύτερες πεντάδες. Αφού αναδειχτούν οι νικητές και μοιραστούν οι μάρκες συμφώνα με την διαδικασία που περιγράφεται στην προηγούμενη παράγραφο, ξεκινά η διαδικασία εκκαθάρισης της παρτίδας. Αρχικά απομακρύνονται όλα τα γραφικά στοιχεία που υπάρχουν στο τραπέζι. Έπειτα αρχικοποιούνται όλες οι μεταβλητές που αφορούν στην εφαρμογή των κανόνων του παιχνιδιού. Τέλος η ροή ελέγχου του συστήματος μεταφέρεται στο βρόχο για τον έλεγχο για νέα εισερχόμενα μηνύματα ή εκκίνηση νέας παρτίδας.

4.2.4 Εφαρμογή πελάτη



Σχήμα 4.2-3: Διάγραμμα δραστηριοτήτων εφαρμογής πελάτη

Στην παραπάνω εικόνα απεικονίζεται το διάγραμμα δραστηριοτήτων της εφαρμογής πελάτη. Η ροή ελέγχου του προγράμματος αρχίζει με την εκκίνηση της εφαρμογής και την εμφάνιση της οθόνης υποδοχής. Αφού εμφανιστεί το μήνυμα με τον σκοπό της υλοποίησης της εφαρμογής στον χρήστη. Το παράθυρο με την οθόνη υποδοχής κλείνει και εμφανίζεται το παράθυρο σύνδεσης του χρήστη.

Στο παράθυρο σύνδεσης εμφανίζεται η κατάσταση των εξυπηρετητών. Για κάθε μεταβολή στην κατάσταση των εξυπηρετητών η εφαρμογής ενημερώνεται αυτόματα για όλη την διάρκεια για την όποια βρίσκεται στην οθόνη σύνδεσης. Όταν ληφθούν δεδομένα για αλλαγή κατάστασης, η κατάλληλη μέθοδος αναλαμβάνει την απεικόνιση της στην αντίστοιχη λεζάντα. Στην περίπτωση που οι εξυπηρετητές είναι ενεργοί ο χρήστης μπορεί να συνδεθεί. Για τη σύνδεση του απαιτείται η συμπλήρωση δυο πεδίων με το ψευδώνυμο και το συνθηματικό του. Κατά την εισαγωγή των στοιχείων του ο χρήστης μπορεί να επιλέξει χρησιμοποιώντας το πεδίο επιλογής έτσι ώστε η εφαρμογή να αποθηκεύσει το ψευδώνυμο του, ώστε να το «θυμάται» στην επομένη εκκίνηση της εφαρμογής. Εφ' όσον ο χρήστης εισάγει τα στοιχεία και δώσει εντολή ξεκινά η διαδικασία σύνδεσης. Αν τα στοιχεία του χρήστη είναι σωστά και επαληθευτεί η σύνδεση, τότε κλείνει το παράθυρο σύνδεσης και εμφανίζεται η κεντρική οθόνη.

Μετά την εμφάνιση της κεντρικής οθόνης ο χρήστης μπορεί να βρει στην κάτω αριστερή γωνία την γραφική κονσόλα επικοινωνίας που χρησιμοποιείται για ανταλλαγή μηνυμάτων με του υπόλοιπους συνδεμένους χρήστες της εφαρμογής. Αφού πληκτρολογηθεί ένα μήνυμα στην κονσόλα από κάποιον χρήστη, αυτό το μήνυμα μεταφέρετε στον κεντρικό εξυπηρετητή και αυτός το προωθεί στου συνδεμένους χρήστες του συστήματος. Όταν ένα μήνυμα φτάσει στην εφαρμογή του πελάτη, τότε «πυροδοτείται» ο κατάλληλος χειριστής, ο όποιος αναλαμβάνει την εμφάνιση του στην γραφική κονσόλα επικοινωνίας.

Στην κεντρική οθόνη εμφανίζεται στο κέντρο του παράθυρου ο πίνακας με τα ενεργά τραπέζια. Στον πίνακα εμφανίζονται οι πληροφορίες με το όνομα του τραπεζιού, τους μέγιστους παίκτες που μπορεί να φιλοξενήσει, καθώς και τους ενεργούς παίκτες την δεδομένη χρονική στιγμή. Όταν σε κάποιο από τα ενεργά τραπέζια παρουσιαστεί μια αλλαγή τότε αυτή μεταφέρεται μέσω του κεντρικού εξυπηρετητή σε όλους τους συνδεμένους χρήστες. Όταν η εφαρμογή πελάτη λάβει στοιχεία του αφορούν σε μεταβολές των τραπεζιών, ενεργοποιείται η κατάλληλη μέθοδος για την αλλαγή των στοιχείων στον πίνακα που εμφανίζεται στην κεντρική οθόνη-λόμπι. Στα αριστερά του πίνακα υπάρχει ένα κουμπί για κάθε τραπέζι που επιτρέπει στους χρήστες να συνδεθούν σε αυτό. Όταν ο χρήστης επιλέξει να συνδεθεί σε ένα από τα ενεργά τραπέζια, ξεκινά η διαδικασία μετάβασης από την οθόνη υποδοχής σε αυτήν του τραπεζιού. Και οι δυο οθόνες φιλοξενούνται από το ίδιο παράθυρο.

Αφού ολοκληρωθεί η μετάβαση από την κεντρική οθόνη στην οθόνη του τραπεζιού, ο χρήστης μπορεί να χρησιμοποιήσει την γραφική κονσόλα επικοινωνίας, η όποια έχει διατηρηθεί και βρίσκεται στο ίδιο σημείο με την κεντρική οθόνη. Στο κάτω και δεξί μέρος του παράθυρου βρίσκεται ο χώρος στον όποιο εμφανίζονται τα γραφικά στοιχεία που χρησιμοποιούνται κατά την διάρκεια του πονταρίσματος από τον παίκτη. Το επάνω μισό τμήμα του παράθυρου φιλοξένει το τραπέζι και της θέσεις αυτό. Κάθε ελεύθερη θέση στο τραπέζι αντιπροσωπεύεται με ένα οβάλ κουμπί, το όποιο φέρει το όνομα «κάθισε εδώ». Όταν ένας παίκτης πατήσει το κουμπί, ξεκινά η διαδικασία

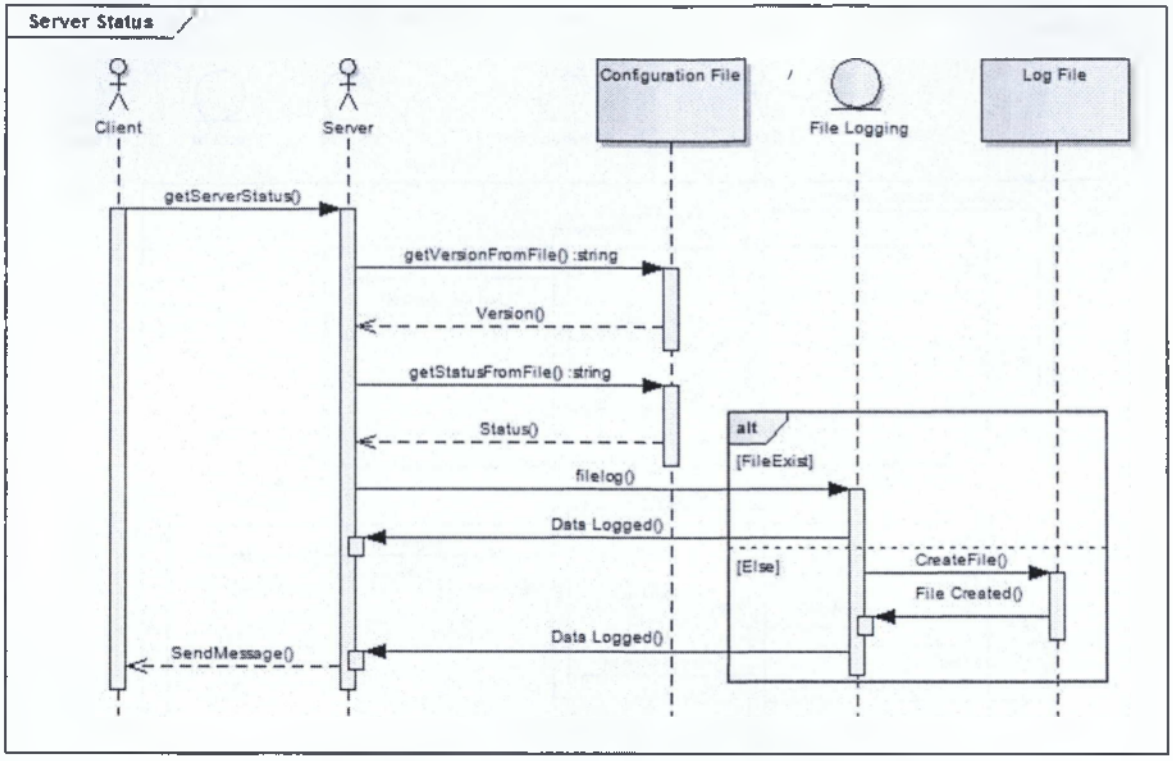
απόκτησης της συγκεκριμένης θέσης από αυτόν. Αφού ο εξυπηρετητής κάνει τους απαραιτήτους ελέγχους δίνει την θέση στον παίκτη εφόσον αυτό είναι εφικτό.

Τέλος όταν στον πελάτη φτάσουν δεδομένα που αφορούν στην εκτέλεση των κανόνων του παιχνιδιού, η ροή ελέγχου του προγράμματος οδηγείται στο κατάλληλο «μονοπάτι», ανάλογα με τον τύπο των δεδομένων του μηνύματος. Αν τα δεδομένα του μηνύματος αφορούν στην τοποθέτηση των τυφλών πονταρισμάτων, τότε καλείται ο κατάλληλος χειριστής, ο όποιος αναλαμβάνει την διεκπεραίωση αυτής της διεργασίας. Αν το μήνυμα περιέχει δεδομένα που αφορούν στα φύλλα των παικτών, τότε «πυροδοτείται» ο κατάλληλος χειριστής για τη διαχείριση αυτών των δεδομένων. Στην περίπτωση που ένας παίκτης πρέπει να κληθεί να ποντάρει τότε σε αυτόν δίνεται η εντολή για ποντάρισμα, και η κατάλληλη μέθοδος «ξεκλειδώνει» τις επιτρεπόμενες για αυτό το ποντάρισμα επιλογές. Όταν ένας παίκτης πραγματοποιήσει ένα ποντάρισμα όλοι οι υπόλοιποι παίκτες που συμμετέχουν και παρακολουθούν την παρτίδα, λαμβάνουν τα δεδομένα με την κίνηση του παίκτη που έκανε την επιλογή του πονταρίσματος. Επιπλέον λαμβάνονται δεδομένα με πληροφορίες σχετικά με τα κοινά φύλλα από τον εξυπηρετητή παιχνιδιού και διαχειρίζονται κατάλληλα από τον αντίστοιχο χειριστή. Τελικά όταν η παρτίδα φτάσει στο τέλος της ο εξυπηρετητής στέλνει τα δεδομένα με τους νικητές και ο χειριστής της εφαρμογής του πελάτη αναλαμβάνει την γραφική απεικόνιση των νικητών και των μεταβολών των εικονικών χρημάτων.

4.3 Διαγράμματα Ακολουθίας

Τα διαγράμματα ακολουθίας παρουσιάζουν τον τρόπο που διαφορετικά αντικείμενα συνεργάζονται μεταξύ τους σε μια χρονική ακολουθία. Συγκεκριμένα παρουσιάζει τα στιγμιότυπα που συμμετέχουν στην αλληλεπίδραση με τις «γραμμές ζωής» τους και τα μηνύματα που ανταλλάσσουν τοποθετημένα στην χρονική ακολουθία.

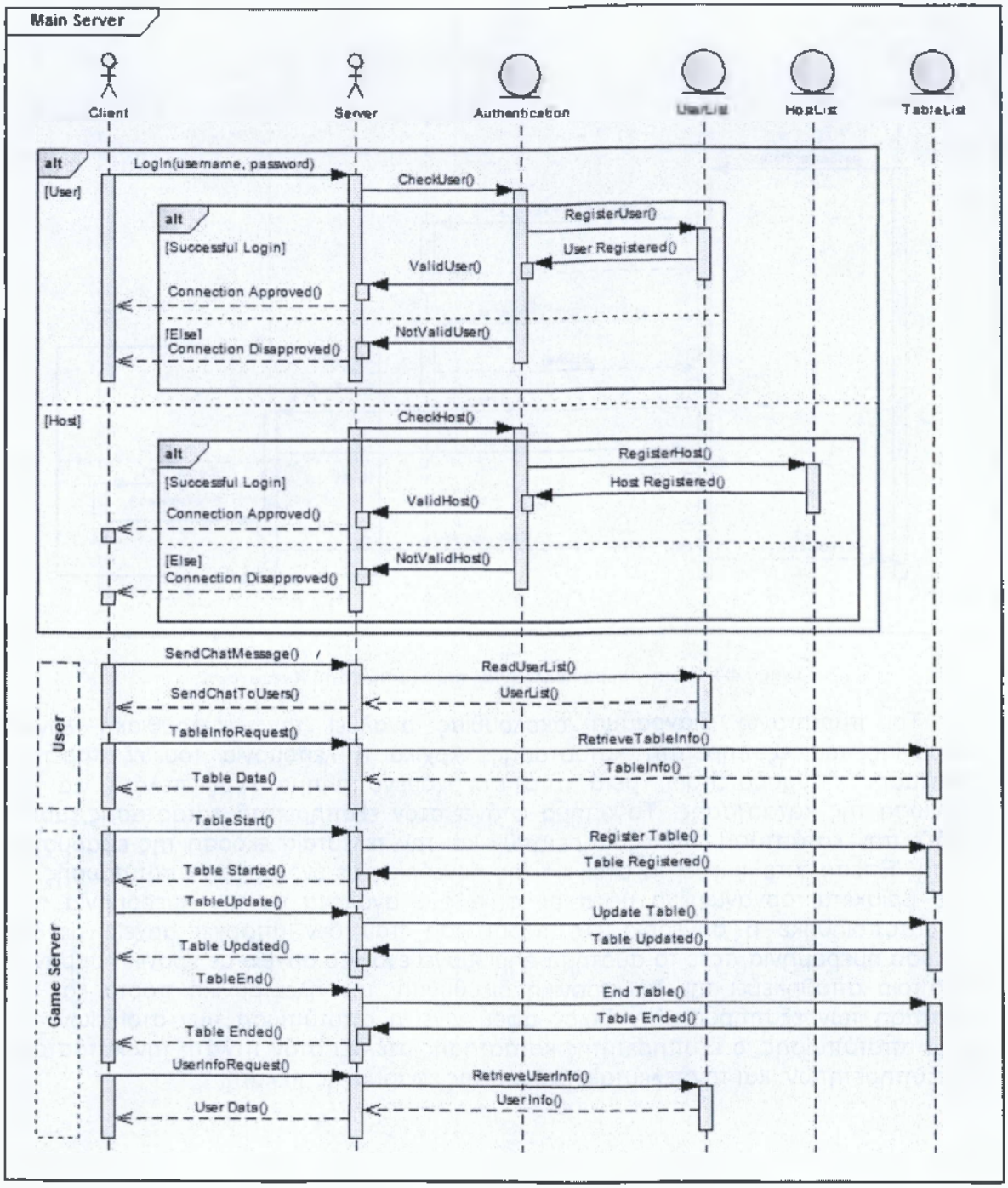
4.3.1 Εξυπηρετητής κατάστασης



Σχήμα 4.3-1: Διάγραμμα ακολουθίας του Εξυπηρετητή Κατάστασης

Το παραπάνω διάγραμμα ακολουθίας αναλύει την ακολουθιακή λογική λειτουργίας του εξυπηρετητή κατάστασης. Αρχικά η λειτουργία του εξυπηρετητή κατάστασης «πυροδοτείται», μετά από εισερχόμενο αίτημα ενός πελάτη για την ανάγνωση της κατάστασης. Το αίτημα φτάνει στον εξυπηρετητή κατάστασης, αυτός διαβάζει την κατάσταση των εξυπηρετητών και την τελευταία έκδοση της εφαρμογής πελάτη. Έπειτα αποθηκεύει τα στοιχεία της συνεδρίας σε ένα αρχείο αποτύπωσης, το όποιο βρίσκεται οργανωμένο μέσα σε φάκελους ανάλογα με την ημερομηνία που πραγματοποιήθηκε η συνέδρια. Σε περίπτωση που δεν υπάρχει αρχείο με την τρέχουσα ημερομηνία, τότε το σύστημα δημιουργεί ένα νέο αρχείο με χρονική σήμανση στο όποιο αποθηκεύει την ηλεκτρονική διεύθυνση, την ηλεκτρονική πόρτα και την κατάσταση των εξυπηρετητών. Τέλος αφού γίνει η αποτύπωση των στοιχείων στο αρχείο αποτύπωσης, ο εξυπηρετητής κατάστασης στέλνει στον πελάτη την κατάσταση των εξυπηρετητών και την τελευταία έκδοση της εφαρμογής πελάτη.

4.3.2 Κεντρικός εξυπηρετητής



Σχήμα 4.3-2: Διάγραμμα ακολουθίας του Κεντρικού Εξυπηρετητή

Στο διάγραμμα που παρουσιάζεται παραπάνω εμφανίζονται όλες οι αλληλεπιδράσεις μεταξύ των αντικειμένων του κεντρικού εξυπηρετητή. Ο πελάτης επιχειρεί να συνδεθεί στον κεντρικό εξυπηρετητή και να αποκτήσει πρόσβαση στα δεδομένα του. Για την σύνδεση του στον κεντρικό εξυπηρετητή ο πελάτης αποστέλλει σε αυτόν το ψευδώνυμο και το συνθηματικό του.

Έπειτα ο κεντρικός εξυπηρετητής ελέγχει τον τύπο του πελάτη. Στην περίπτωση που ο πελάτης είναι απλός χρήστης, εκκινείται η διαδικασία σύνδεσης αυτού. Γίνεται έλεγχος για την ορθότητα των στοιχείων του, και εφ' όσον αυτό επιβεβαιωθεί από το σύστημα, τα στοιχεία του χρήστη εγγράφονται στην αντίστοιχη λίστα. Τέλος εγκρίνεται η σύνδεση αυτού στον κεντρικό εξυπηρετητή. Διαφορετικά η σύνδεση του χρήστη στο σύστημα απορρίπτεται.

Για την περίπτωση που ο πελάτης είναι εξυπηρετητής παιχνιδιού, ο κεντρικός εξυπηρετητής διενεργεί τους απαραίτητους ελέγχους. Αφού διαπιστωθεί τόσο η ορθότητα των στοιχείων σύνδεσης, όσο και της ικανότητας του εξυπηρετητή παιχνιδιού να φιλοξενήσει κάποιο από τα τραπέζια του παιχνιδιού, τότε ο κεντρικός εξυπηρετητής αποδέχεται την αίτηση σύνδεσης και παρέχει στον εξυπηρετητή παιχνιδιού πρόσβαση στα δεδομένα του συστήματος. Σε διαφορετική περίπτωση γίνεται άρνηση πρόσβασης στον κεντρικό εξυπηρετητή εμφανίζοντας στον χρήστη το αντίστοιχο μήνυμα.

Όταν ένας χρήστης που έχει συνδεθεί στο σύστημα επιθυμεί να στείλει ένα μήνυμα προς τους χρήστες της εφαρμογής, ο χρήστης συντάσσει και στέλνει το μήνυμα στον κεντρικό εξυπηρετητή. Ο κεντρικός εξυπηρετητής διαβάζει την λίστα των συνδεμένων χρηστών και προωθεί το μήνυμα σε αυτούς.

Στην περίπτωση που κάποιος χρήστης επιθυμεί να λάβει τη λίστα με τις πληροφορίες των ενεργών τραπεζιών, ο χρήστης αυτός στέλνει ένα μήνυμα με το αίτημα του στον κεντρικός εξυπηρετητή. Ο εξυπηρετητής διαβάζει τα δεδομένα από την λίστα με της πληροφορίες των τραπεζιών και τις στέλνει στον χρήστη.

Για την εκκίνηση ενός τραπεζιού, ο πελάτης - εξυπηρετητής παιχνιδιού στέλνει στον κεντρικό εξυπηρετητή αίτημα για την εκκίνηση του συγκεκριμένου τραπεζιού. Ο κεντρικός εξυπηρετητής αφού ενημερώσει τα στοιχεία της λίστα των τραπεζιών, με τις πληροφορίες του νέου τραπεζιού, δίνει εντολή στον εξυπηρετητή παιχνιδιού για την εκκίνηση της φιλοξενίας του τραπεζιού. Τέλος ο κεντρικός εξυπηρετητής ενημερώνει τους συνδεμένους χρήστες για την ύπαρξη νέου τραπεζιού.

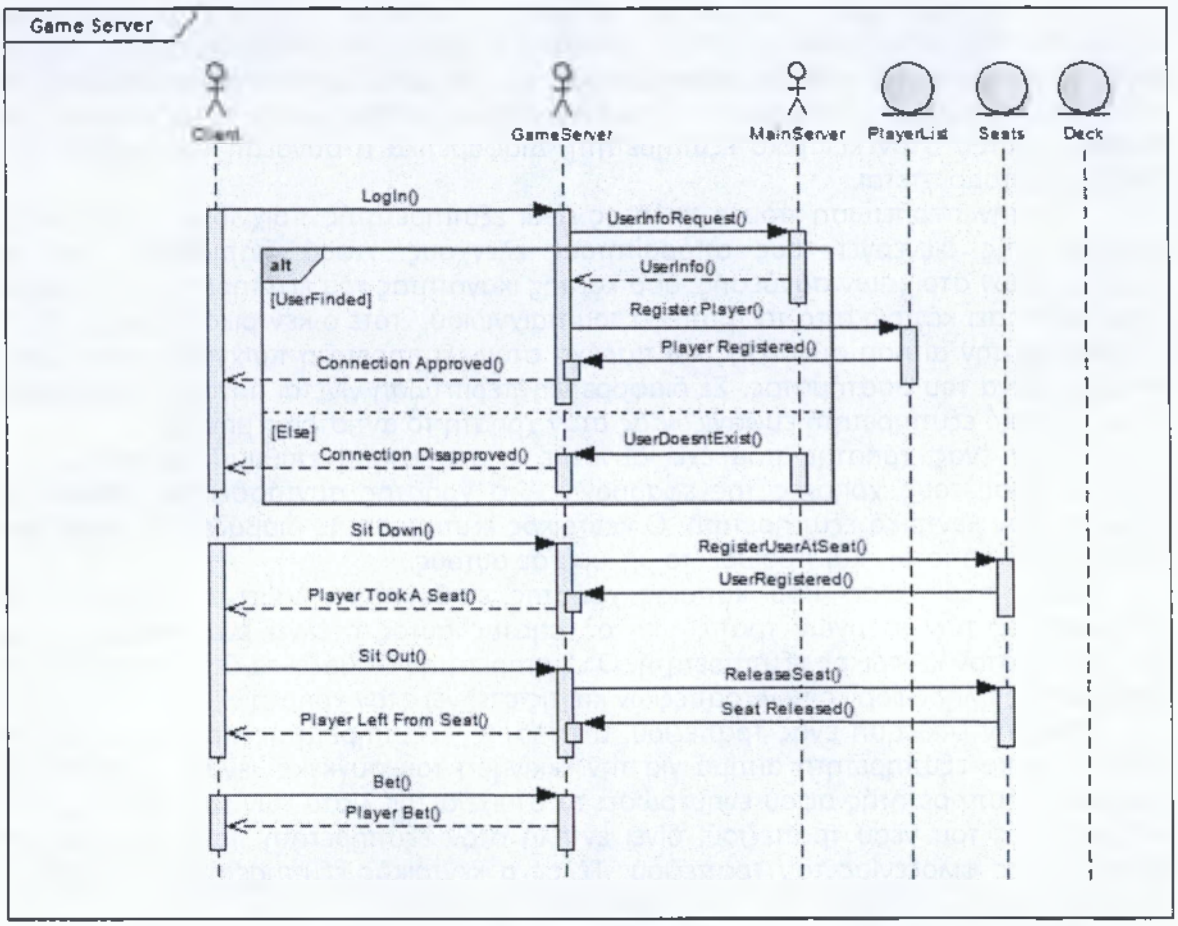
Όταν παρουσιάζεται μια μεταβολή στα δεδομένα του τραπεζιού που φιλοξενείται από έναν εξυπηρετητή παιχνιδιού, τότε ο εξυπηρετητής παιχνιδιού στέλνει τα δεδομένα των μεταβολών στον κεντρικό εξυπηρετητή. Έπειτα, ο κεντρικός εξυπηρετητής ανανεώνει τις πληροφορίες της λίστας των τραπεζιών, με τα νέα δεδομένα. Τέλος τα νέα δεδομένα στέλνονται στους συνδεμένους χρήστες του συστήματος.

Κατά την αποσύνδεση ενός εξυπηρετητή παιχνιδιού από το σύστημα, στέλνεται από τον εξυπηρετητή παιχνιδιού, μήνυμα με το αίτημα κλεισίματος του τραπεζιού. Ο κεντρικός εξυπηρετητής λαμβάνει το νέο μήνυμα. Καταργεί τις πληροφορίες του τραπεζιού από την λίστα με τα ενεργά τραπέζια. Τέλος ενημερώνει τους χρήστες για την κατάργηση του συγκεκριμένου τραπεζιού.

Κάθε φορά που ο εξυπηρετητής χρειάζεται τα στοιχεία ενός χρήστη, στέλνει μήνυμα στον κεντρικό εξυπηρετητή ζητώντας αυτές τις πληροφορίες. Ο κεντρικός εξυπηρετητής διαβάζει τα δεδομένα από την λίστα των χρηστών. Τέλος ο κεντρικός

εξυπηρετητής στέλνει ένα μήνυμα με τα στοιχεία του χρήστη στον εξυπηρετητή παιχνιδιού.

4.3.3 Εξυπηρετητής παιχνιδιού



Σχήμα 4.3-3: Διάγραμμα ακολουθίας των μηνυμάτων επικοινωνίας του Εξυπηρετητή Παιχνιδιού

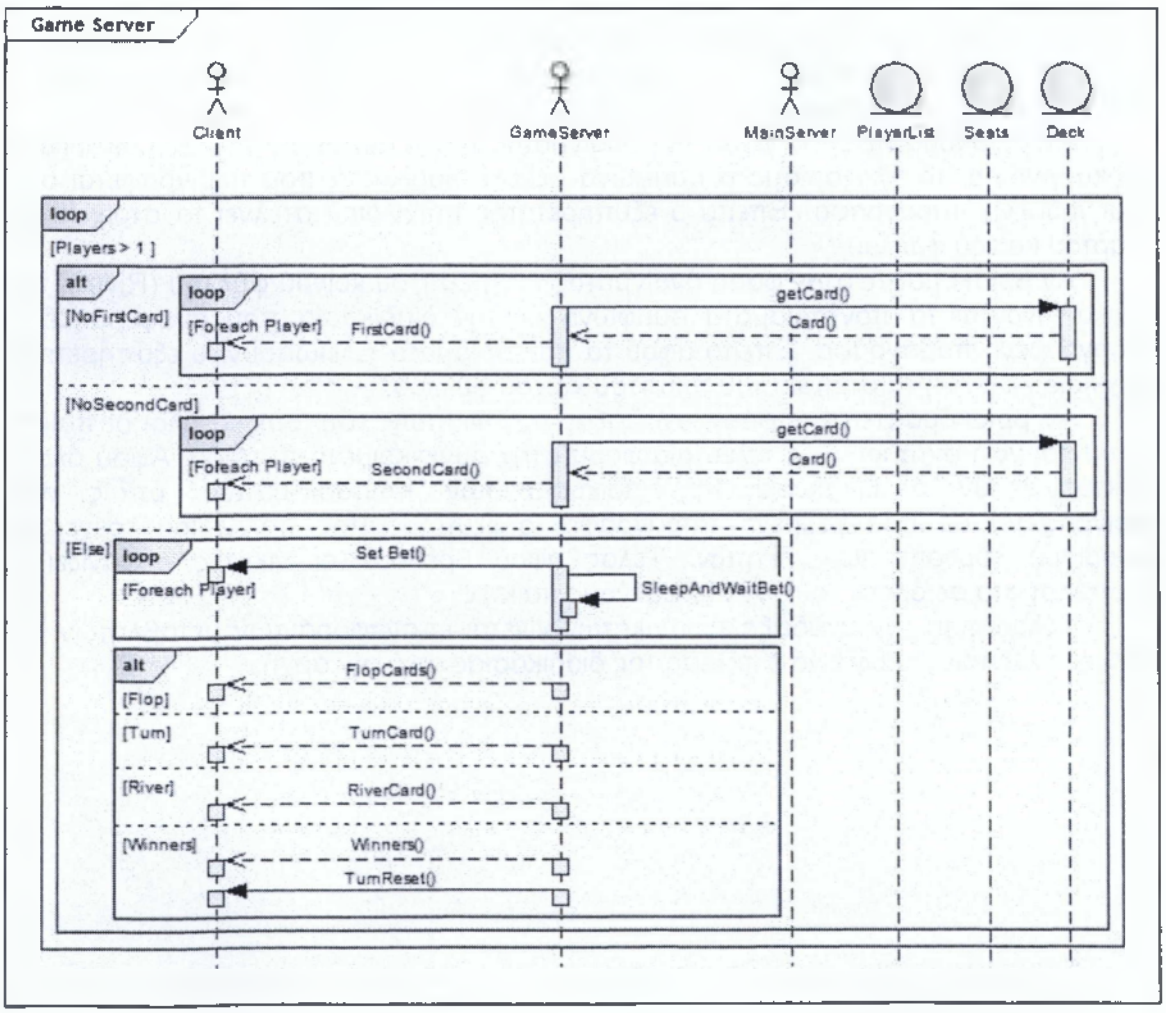
Στο παραπάνω διάγραμμα όταν ένας παίκτης θέλει να συνδεθεί στον εξυπηρετητή παιχνιδιού, στέλνει αίτηση σύνδεσης στον εξυπηρετητή παιχνιδιού. Αυτός με την σειρά του στέλνει τα στοιχεία του παίκτη στον κεντρικό εξυπηρετητή για την διασταύρωση τους με αυτά της λίστας των συνδεμένων χρηστών. Στην περίπτωση που ο χρήστης εντοπιστεί ανάμεσα στους συνδεμένους χρήστες, ο κεντρικός εξυπηρετητής στέλνει τα στοιχεία που απαιτούνται στον εξυπηρετητή παιχνιδιού. Διαφορετικά αν ο χρήστης δεν εντοπιστεί μέσα στην λίστα με τους συνδεμένους χρήστες, απορρίπτεται η σύνδεση του στον εξυπηρετητή παιχνιδιού.

Όταν ένας παίκτης συνδεμένος στον εξυπηρετητή παιχνιδιού θελήσει να καθίσει σε μια θέση του τραπεζιού. Ο παίκτης στέλνει μήνυμα στον εξυπηρετητή. Έπειτα ο εξυπηρετητής διαβάζει το αίτημα του χρήστη και περνά τα στοιχεία του χρήστη στην

λίστα με τις θέσεις. Τέλος ανακοινώνει σε όλους τους συνδεμένους σε αυτόν χρήστες την απόκτηση της θέσης από τον συγκεκριμένο παίκτη.

Ας υποθέσουμε ότι κάποια στιγμή κάποιος από τους παίκτες που κάθονται στις θέσεις του τραπεζιού θελήσει να αποχώρηση από την θέση του. Για την έναρξη της διαδικασίας στέλνει αίτημα στον εξυπηρετητή παιχνιδιού και των ενημερώνει για την αποχώρηση του από την θέση του. Ο εξυπηρετείς παιχνιδιού αποδεσμεύει τον παίκτη από αυτήν την θέση. Τέλος ο εξυπηρετητής στέλνει μήνυμα σε όλους τους συνδεδεμένους σε αυτόν παίκτες και τους ενημερώνει για την αποχώρηση του παίκτη από την συγκεκριμένη θέση.

Όταν είναι η σειρά ενός παίκτη προκειμένου να ποντάρει, μόλις αυτός ο παίκτης επιλέξει το ποντάρισμα του, η εφαρμογή στέλνει ένα μήνυμα με τα δεδομένα του ποντάρίσματος στον εξυπηρετητή παιχνιδιού. Έπειτα ο εξυπηρετητής παιχνιδιού μετασχηματίζει τα δεδομένα και τα στέλνει σε όλους τους συνδεμένους χρήστες. Ακολουθεί σχήμα στην επόμενη σελίδα.



Σχήμα 4.3-4: Διάγραμμα ακολουθίας των κανόνων παιχνιδιού του Εξυπηρετητή Παιχνιδιού

Στο παραπάνω διάγραμμα ακολουθίας απεικονίζεται η αλληλεπίδραση των αντικειμένων που συσχετίζονται με την υλοποίηση των κανόνων του παιχνιδιού. Αρχικά στο τμήμα του εξυπηρετητή παιχνιδιού τρέχει ένας ατέρμονος βρόχος που ελέγχει τον αριθμό των ενεργών παικτών.

Όταν ο αριθμός των παικτών που κάθονται στο τραπέζι είναι μεγαλύτερος του ενός, ο εξυπηρετητής παιχνιδιού ελέγχει αν όλοι οι παίκτες έχουν την πρώτη τους κάρτα στο χέρι τους. Αν δεν υπάρχει η πρώτη κάρτα στο χέρι κάθε παίκτη, τότε ο εξυπηρετητής παιχνιδιού τραβά μια κάρτα από την τράπουλα και στέλνει τα στοιχεία της στο παίκτη. Αυτή η διαδικασία επαναλαμβάνεται για κάθε παίκτη χωρίς την πρώτη κάρτα στο χέρι του. Έπειτα ακολουθείται η ίδια διαδικασία και για την δεύτερη κάρτα που θα πρέπει να έχουν οι παίκτες στο χέρι τους.

Όταν όλοι οι παίκτες έχουν από δυο κάρτες στο χέρι τους, ο εξυπηρετητής παιχνιδιού ελέγχει σε ποιο σημείο της παρτίδα βρισκόμαστε. Αν βρισκόμαστε στην φάση ανοίγματος των τριών κοινών φύλλων (Flop), τότε ξεκινά το ποντάρισμα για κάθε παίκτη. Ο εξυπηρετητής παιχνιδιού δίνει τον λόγο για ποντάρισμα σε κάθε παίκτη με την σειρά, όπως αυτή ορίζεται από τους κανόνες του παιχνιδιού. Μέχρι ο παίκτης να ποντάρει το σύστημα τίθεται σε αναμονή. Όταν όλοι οι παίκτες ποντάρουν, τότε ο εξυπηρετητής παιχνιδιού στέλνει τα στοιχεία των τριών κοινών φύλλων (Flop).

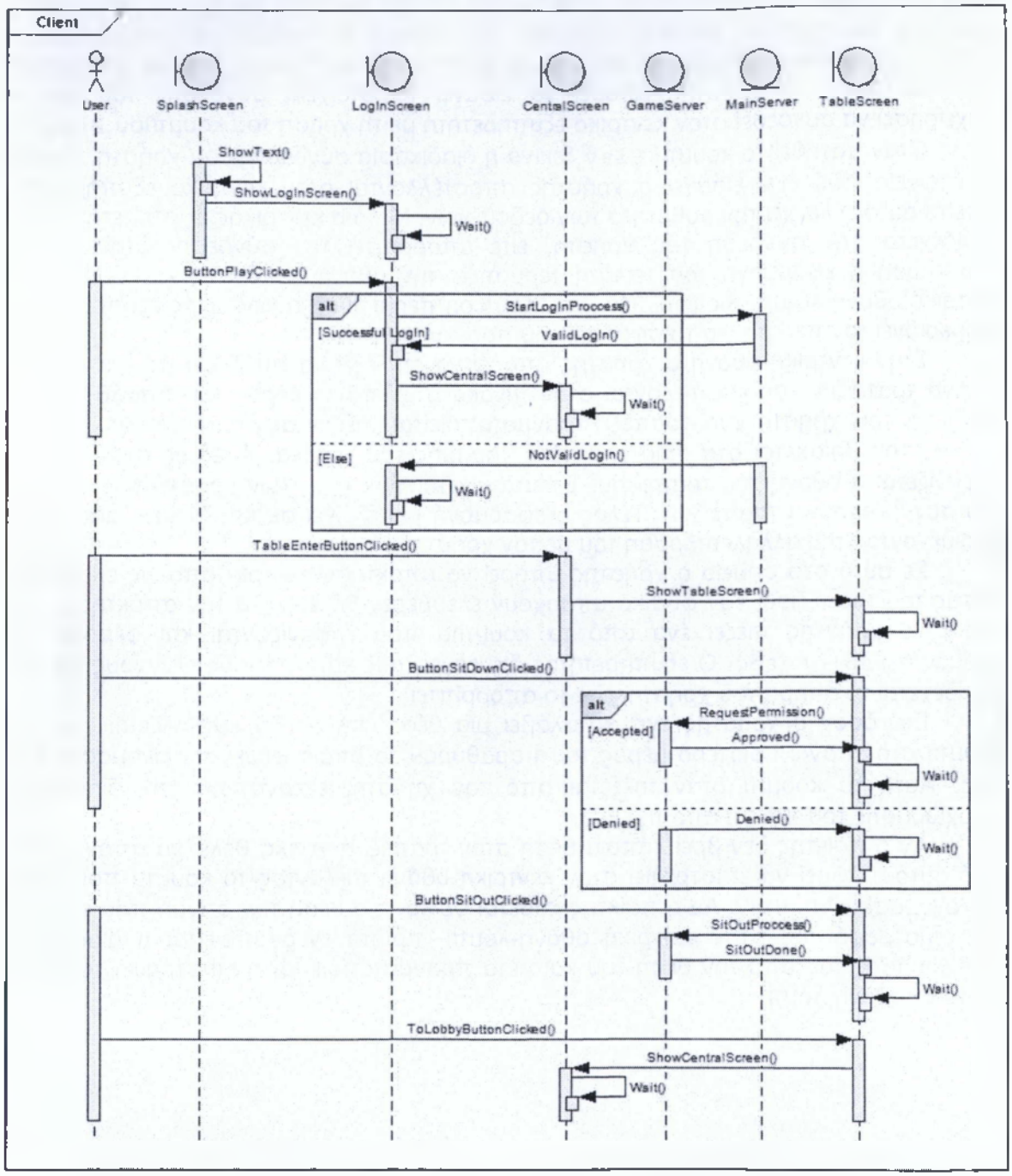
Αν βρισκόμαστε στην φάση του ανοίγματος του τέταρτου κοινού φύλλου (Turn), αρχικά γίνονται τα ποντάρισματα συμφώνα με την διαδικασία που περιγράφεται στην προηγούμενη παράγραφο. Έπειτα ο εξυπηρετητής παιχνιδιού στέλνει τα στοιχεία του τέταρτου κοινού φύλλου.

Αν βρισκόμαστε στην φάση ανοίγματος του πέμπτου κοινού φύλλου (River), τότε αρχικά γίνονται τα ποντάρισματα συμφώνα με την διαδικασία που αναφέρθηκε σε προηγούμενη παράγραφο. Έπειτα αφού τα ποντάρισματα τελειώσουν, ο εξυπηρετητής παιχνιδιού στέλνει τα στοιχεία του πέμπτου κοινού φύλλου.

Αν βρισκόμαστε στην φάση ανάδειξης των νικητών, τότε αρχικά όλοι οι παίκτες καλούνται να ποντάρουν για τελευταία φορά στην συγκεκριμένη παρτίδα. Αφού όλοι οι παίκτες έχουν ολοκληρώσει τη διαδικασία των ποντάρισμάτων, όπως αυτή περιγράφεται σε προηγούμενη παράγραφο, ο εξυπηρετητής παιχνιδιού τρέχει τον αλγόριθμο εύρεσης των νικητών. Τέλος αφού βρεθούν οι νικητές εμφανίζει τα αποτελέσματα σε όλους τους συνδεδεμένους παίκτες.

Τέλος μετά την ανάδειξη των νικητών γίνεται επαναφορά των μεταβλητών στις αρχικές τους τιμές και ξεκινά εκτέλεση της διαδικασίας από την αρχή.

4.3.4 Εφαρμογή πελάτη



Σχήμα 4.3-5: Διάγραμμα ακολουθίας της εφαρμογής πελάτη

Στο παραπάνω διάγραμμα ακολουθίας παρουσιάζεται η σχέση αλληλεπίδρασης των αντικειμένων που συσχετίζονται με την εφαρμογή πελάτη . Το πρόγραμμα ξεκινά

με την εμφάνιση της οθόνης υποδοχής και το κείμενο με τον σκοπό της πτυχιακής. Μετά την εμφάνιση του κειμένου η οθόνη κλείνει και γίνεται η μετάβαση στην οθόνη σύνδεσης του χρήστη. Με την εμφάνιση της οθόνης το πρόγραμμα παραμένει σε κατάσταση αδράνειας περιμένοντας για την πιθανή αλληλεπίδραση του με τον χρήστη της εφαρμογής. Ο χρήστης μπορεί να εισάγει τα στοιχεία σύνδεσης του και να επιχειρήσει να συνδεθεί στον κεντρικό εξυπηρετητή με τη χρήση του κουμπιού `Play`.

Όταν πατηθεί το κουμπί `Play` ξεκινά η διαδικασία σύνδεσης του χρήστη. Αρχικά τα στοιχεία που έχει εισάγει ο χρήστης αποστέλλονται στον κεντρικό εξυπηρετητή. Έπειτα αυτός ελέγχει την ορθότητα των δεδομένων. Μετά ο κεντρικός εξυπηρετητής είτε αποδέχεται την σύνδεση του χρήστη, είτε απορρίπτει την σύνδεση. Στην πρώτη περίπτωση η εφαρμογή του πελάτη τερματίζει την οθόνη σύνδεσης και εκκινεί την κεντρική οθόνη-λόμπι. Διαφορετικά στην δεύτερη περίπτωση ο κεντρικός εξυπηρετητής ενημερώνει τον πελάτη για το σφάλμα που προέκυψε.

Στην κεντρική οθόνη ο χρήστης μπορεί να επιλέξει να συνδεθεί σε ένα από τα ενεργά τραπέζια που εμφανίζονται στον πίνακα στο επάνω μέρος του παράθυρου. Η σύνδεση του χρήστη στο τραπέζι πραγματοποιείται πιέζοντας το αντίστοιχο κουμπί `Enter` που βρίσκεται στα αριστερά κάθε γραμμής του πίνακα. Αμέσως στον χρήστη εμφανίζεται η οθόνη του τραπεζιού. Έπειτα γίνεται φόρτωση των γραφικών στοιχείων του συγκεκριμένου τραπεζιού. Τέλος η εφαρμογή μεταβαίνει σε κατάσταση αδράνειας περιμένοντας την αλληλεπίδραση του με τον χρήστη.

Σε αυτό στο σημείο ο χρήστης μπορεί να αποκτήσει κάποια από τις ελεύθερες θέσεις του τραπεζιού, εάν φυσικά υπάρχουν ελεύθερες θέσεις. Για την απόκτηση μιας θέσης, ο χρήστης πιέζει ένα από τα κουμπιά που εμφανίζονται και φέρουν την ονομασία `κάθισε εδώ`. Ο εξυπηρετητής διενεργεί τους καταλλήλους ελέγχους και είτε αποδέχεται το αίτημα του χρήστη, είτε το απορρίπτει.

Εφ' όσον ο χρήστης έχει καταλάβει μια θέση, πλέον θα εμφανίζεται ένα νέο κουμπί στο επάνω αριστερό μέρος του παράθυρου το όποιο φέρει την ονομασία `Sit Out`. Αυτό το κουμπί όταν πιέζεται από τον χρήστη, ενεργοποιεί την διαδικασία αποχώρησης του παίκτη από την θέση.

Αν ο παίκτης δεν βρει κάποια θέση στον τραπέζι ή γενικά θέλει να αποχωρήσει από αυτό, μπορεί να επιστρέψει στην κεντρική οθόνη πιέζοντας το κουμπί που φέρει την ονομασία `ToLobby`. Αν ο παίκτης κάθεται σε κάποια θέση την στιγμή που επιλεγεί την επιστροφή του στην κεντρική οθόνη-λόμπι, πρώτα ενεργοποιείται η διαδικασία αποχώρησης του από την θέση του και μετά πραγματοποιείται η επιστροφή του στην κεντρική οθόνη-λόμπι.

Επίλογος - Συμπεράσματα

Συμπεράσματα

Φτάνοντας στην ολοκλήρωση της εργασίας μας θα θέλαμε να καταθέσουμε τα συμπεράσματα, τις εμπειρίες και τις γνώσεις μας από την όλη διαδικασία της συγγραφής της εργασίας. Αν και γνωρίζαμε την δυσκολία του θέματος που επιλέξαμε να αναλάβουμε, κατά την διάρκεια της υλοποίησης του θέματος διαπιστώσαμε τον πραγματικά υψηλό βαθμό δυσκολίας του. Για τον λόγο αυτό ο χρόνος που πιστεύαμε ότι θα χρειαζόταν για την ολοκλήρωση της εργασίας ήταν μεγαλύτερος από αυτόν που είχαμε αρχικά εκτιμήσει. Να σημειώσουμε επίσης ότι στην χρονική καθυστέρηση έπαιξαν ρόλο και εξωγενείς παράγοντες.

Σίγουρα οι λειτουργιές καθώς και το επίπεδο ανάπτυξης που θέλαμε να έχει η εφαρμογή μας ήταν διαφορετικά από αυτά που σχεδιάζαμε. Πιστεύουμε πάντως ότι οι γνώσεις, οι εμπειρίες και όλα όσα αποκομίσαμε καθ' όλη την διάρκεια ενασχόλησης με την συγκεκριμένη εργασία ήταν πολύ περισσότερα από αυτά που φανταζόμασταν.

Ο σκοπός της εργασίας μας φυσικά δεν ήταν η υλοποίηση ενός τέλειου συστήματος της αρχιτεκτονικής εξυπηρετητή - πελάτη, αλλά περισσότερο η αξιοποίηση όσο το δυνατό περισσότερων νέων για εμάς τεχνολογιών και εργαλείων. Εξίσου βασικός στόχος της εργασίας ήταν επίσης η ανάπτυξη ενός έργου που θα αποτελέσει την βάση για μελλοντική εξέλιξη του είτε από εμάς τους ίδιους είτε από αλλά άτομα που θα θελήσουν να ασχοληθούν με το συγκεκριμένο αντικείμενο.

Θα θέλαμε σε αυτό το σημείο να πληροφορήσουμε όσους θελήσουν να ασχοληθούν στο μέλλον με την ανάπτυξη ενός πολύπλοκου συστήματος όπως το συγκεκριμένο, ότι θα πρέπει να είναι έτοιμοι να αντιμετωπίσουν σημαντικά εμπόδια στην πορεία τους. Θα πρέπει να κάνουν μεγάλη προσπάθεια και να αφιερώσουν πολύ χρόνο στο να μελετήσουν το υλικό που υπάρχει ώστε να φτάσουν σε σημείο να ξεπεράσουν τα εξειδικευμένα προγραμματιστικά προβλήματα που θα συναντήσουν. Στο τέλος βέβαια θεωρούμε ότι θα κερδίσουν σημαντικά εφόδια και εμπειρίες δημιουργώντας κάτι που θα τους δώσει ιδιαίτερη ικανοποίηση.

Κατά τον σχεδιασμό της εφαρμογής, υπήρξαν αρκετές σκέψεις για τις λειτουργίες που θα μπορούσε να έχει η εφαρμογή μας στην τελική της μορφή. Στο συγκεκριμένο έγγραφο έχουν αναφερθεί σε προηγούμενα κεφάλαια οι τομείς στους οποίους μπορεί να αναπτυχτεί περαιτέρω η εφαρμογή, όπως για παράδειγμα η ασφάλεια και η βελτιστοποίηση της εφαρμογής. Πέρα από την βελτίωση στους προαναφερθέντες τομείς, θα μπορούσαν να προστεθούν λειτουργίες που θα επιτρέπουν στους χρήστες την δημιουργία και διαχείριση των λογαριασμών τους. Επίσης θα μπορούσε να αναπτυχτεί μια μέθοδος για την αναβάθμιση της εφαρμογής κατά την εκκίνηση της. Ακόμα θα μπορούσε να αναπτυχτεί μια μέθοδος για την καταγραφή των ενεργειών και των σφαλμάτων που προκύπτουν τόσο στην πλευρά των εξυπηρετητών όσο και σ' αυτή των πελατών. Επιπρόσθετα θα μπορούσαν να προστεθούν ηχητικά εφέ στην εφαρμογή πελατών για την επένδυση των κινήσεων. Τέλος καλό θα ήταν να γίνει μια καλύτερη οργάνωση του κώδικα και των μηνυμάτων που παρουσιάζονται στους χρήστες της εφαρμογής πελάτη και των εφαρμογών των εξυπηρετητών.

Πήγες και βιβλιογραφία

[1] Neoforce, "Neoforce"
http://neoforce.codeplex.com/ (Ανάκτηση Νοέμβριος 05, 2013).

[2] Sparx Systems, "Enterprise Architect UML Tutorial"
http://www.sparxsystems.com.au/uml-tutorial.html (Ανάκτηση Νοέμβριος 11, 2012).

[3] Wikepedia,"Peer to peer"
http://en.wikipedia.org/wiki/Peer-to-peer (Ανάκτηση Νοέμβριος 11, 2012).

[4] Wikipedia, "Betting in poker"
http://en.wikipedia.org/wiki/Betting_(poker) (Ανάκτηση Απρίλιος 10, 2013).

[5] Wikipedia, "List of poker hands"
http://en.wikipedia.org/wiki/List_of_poker_hands (Ανάκτηση Απρίλιος 10, 2013).

[6] Wikipedia, "Common Intermediate Language"
http://en.wikipedia.org/wiki/Common_Intermediate_Language (Ανάκτηση Απρίλιος 06, 2013).

[7] Wikipedia, "Common Language Infrastructure"
http://en.wikipedia.org/wiki/Common_Language_Infrastructure (Ανάκτηση Απρίλιος 05, 2013).

[8] Wikipedia, "Microsoft XNA"
http://en.wikipedia.org/wiki/Microsoft_XNA (Ανάκτηση Νοέμβριος 02, 2012).

[9] Wikipedia, "Application programming interface"
http://en.wikipedia.org/wiki/Application_programming_interface (Ανάκτηση Νοέμβριος 02, 2012).

[10] Wikipedia, "End User License Agreement"
http://en.wikipedia.org/wiki/End-user_license_agreement (Ανάκτηση Νοέμβριος 05, 2012).

[11] Wikipedia, "SharpDX"
http://en.wikipedia.org/wiki/SharpDX (Ανάκτηση Απρίλιος 16, 2013).

[12] Wikipedia, "Unified Modeling Language"
http://en.wikipedia.org/wiki/Unified_Modeling_Language (Ανάκτηση Απρίλιος 15, 2013).

[13] MSDN, "Community Technology Preview" http://blogs.msdn.com/b/sqlphp/archive/2010/06/14/what-is-a-community-technology-preview-ctp.aspx (Ανάκτηση Νοέμβριος 03, 2012).

[14] Mike Kukta's Blog, "Neoforce Tutorial" http://www.mikekukta.com/post/2011/04/10/Neoforce-Tutorial-Easy-Setup-.aspx (Ανάκτηση Νοέμβριος 08, 2012).

[15] Lidgren, M, " Lidgren Network Gen3" http://code.google.com/p/lidgren-network-gen3/ (Ανάκτηση Νοέμβριος 09, 2012).

[16] Reed, A., "Learning XNA 4.0: Game Development for the PC, Xbox", O'Reilly Media, 2010.

[17] Sharp, J., "Microsoft(R) Visual C#(R) 2010 Step by Step", Microsoft Press, 2010.

[18] Wikipedia, "Texas hold 'em", http://en.wikipedia.org/wiki/Texas_hold_'em (Ανάκτηση Απρίλιος 10, 2013).

[19] Videogames Laboratory, "Εισαγωγή στην C#/.NET", http://videogameslab.wordpress.com/2009/01/08/intro-csharp-net/ (Ανάκτηση Απρίλιος 07, 2013)

[20] Κώτης, Κ, "ΣΗΜΕΙΩΣΕΙΣ UML στα Ελληνικά", 2006, http://www.icsd.aegean.gr/kotis/softTech06/UMLnotes.PDF

[21] Τζίτζικας , Γ, "Ανάλυση και Σχεδίαση Πληροφοριακών Συστημάτων", 2007, http://www.csd.uoc.gr/~hy351/2007/downloads/assisting_lectures/presentations/I S_351_Requirements_DFD_ProcessDescr_2007_08.pdf

[22] Τζίτζικας , Γ, "Περίπτωση Χρήσης", 2007, http://www.csd.uoc.gr/~hy351/2007/downloads/assisting_lectures/presentations/I S_351_UseCases_2007_11.pdf

Παράρτημα - Εργαλεία Υλοποίησης & Οδηγίες Εγκατάστασης

Η ανάπτυξη των προγραμμάτων του συστήματος έγινε με την χρήση του ολοκληρωμένου περιβάλλοντος ανάπτυξης (IDE) Microsoft Virtual Studio 2010. Λόγω της ιδιαιτερότητας της εφαρμογής πελάτη που αναπτύξαμε χρησιμοποιήσαμε το XNA Game Studio ως πρόσθετο στο ολοκληρωμένο περιβάλλον ανάπτυξης του Visual Studio. Με την χρήση του XNA Game Studio είχαμε την δυνατότητα να υλοποιήσουμε την εφαρμογή του πελάτη χρησιμοποιώντας μεθοδολογίες που ταιριάζουν περισσότερο σε μια εφαρμογή παιχνιδιού. Επίσης για την λειτουργία της βάσης δεδομένων που απαιτείται από την κεντρική εφαρμογή χρησιμοποιήσαμε το Microsoft SQL Server 2008 R2.

Όλα τα διαγράμματα UML που δημιουργήθηκαν και εμφανίζονται στο πλαίσιο της παρούσας εργασίας υλοποιηθήκαν με την χρήση της εφαρμογής Enterprise Architect της εταιρίας Spar Systems. Όλα τα διαγράμματα υλοποιήθηκαν με γνώμονα την ευκολότερη ανάλυση των λειτουργιών του συστήματος κι ενδεχομένως να μην υπάρχει πλήρης ταύτιση των διαγραμμάτων με τον κώδικα.

Για την λειτουργία της εφαρμογής πελάτη είναι απαραίτητη η εγκατάσταση των παρακάτω προγραμμάτων:

1) .NET Framework 4 Full
2) XNA Framework Redistributable

Και τα δυο βρίσκονται στον φάκελο "\Poker\Programs\Requirements" του οπτικού δίσκου που συνοδεύει το παρόν έγγραφο.

Για το άνοιγμα και την επεξεργασία του πλήρους έργου απαιτούνται οι παρακάτω ενέργειες:

1) Ύπαρξη Ολοκληρωμένου υπερβάλλοντος ανάπτυξης (IDE)
2) Ύπαρξη XNA Game Studio

Το δεύτερο πρόγραμμα εγκατάστασης υπάρχει στον οπτικό δίσκο που συνοδεύει το παρόν κείμενο και συγκεκριμένα στον φάκελο "\Poker\Programs\Requirements".

Στον οπτικό δίσκο που συνοδεύει το παρόν κείμενο υπάρχουν μέσα στον φάκελο "Programs" δυο έτοιμες ομάδες των προγραμμάτων πελάτη και εξυπηρετητών. Στον φάκελο "Local Host" υπάρχουν τα προγράμματα του πελάτη και των εξυπηρετητών που εκτελούνται στο εσωτερικό δίκτυο του υπολογιστή. Ενώ στον φάκελο "Internet" υπάρχουν τα προγράμματα που εκτελούνται στο εξωτερικό δίκτυο με την χρήση δυναμικής διεύθυνσης διαδικτύου. Για περισσότερες πληροφορίες σχετικά με τις πλήρεις διαδρομές των προγραμμάτων μπορείτε να διαβάσετε το αρχείο κειμένου με τίτλο "Notes" που βρίσκεται μέσα στον οπτικό δίσκο.

Για την χρήση των εφαρμογών πελάτη και εξυπηρετητών που έχουν δημιουργηθεί για την χρήση στο εξωτερικό δίκτυο -internet- μέσω DDNS έχει δημιουργηθεί ένας λογαριασμός στην σελίδα www.dnsexit.com. Με την χρήση των DDNS και του client που υπάρχει μέσα στα προγράμματα του οπτικού δίσκου μπορούν

να εκτελεστούν τα προγράμματα των εξυπηρετητών πάνω σε δυναμικές ηλεκτρονικές διευθύνσεις χωρίς να υπάρχει πρόβλημα στην επικοινωνία. Περισσότερες λεπτομέρειες υπάρχουν στο αρχείο κειμένου με τίτλο "**Notes**" που βρίσκεται μέσα στον οπτικό δίσκο.

Στον οπτικό δίσκο που συνοδεύει το συγκεκριμένο κείμενο υπάρχει μέσα σε αρχείο SQL ο κώδικας για την δημιουργία της βάσης δεδομένων που είναι απαραίτητη για την λειτουργία του κεντρικού εξυπηρετητή. Η βάση πρέπει να βρίσκεται εγκατεστημένη στο ίδιο σύστημα με αυτό του κεντρικού εξυπηρετητή.

Το συγκεκριμένο έργο υλοποιήθηκε για εκπαιδευτικούς σκοπούς στα πλαίσια της πτυχιακής εργασίας του τμήματος «Τεχνολογίας Πληροφορικής και Τηλεπικοινωνιών» του ΤΕΙ Καλαμάτας. Όσοι επιθυμούν μπορούν να χρησιμοποιήσουν εξ ολοκλήρου ή τμήμα του έργου μας, ενώ μπορούν να προχωρήσουν και στην ανάπτυξη αυτής για δικούς τους σκοπούς. Σε κάθε περίπτωση χρήσης θα πρέπει να υπάρχει η ανάλογη αναγνώριση προς το συγκεκριμένο έργο για τα στοιχεία που προέρχονται μέσα από αυτό.