

School: אורט ע"ש נעמי שמר גן יבנה

Project name: Battle Ship

Student’s name: Ariel Ohana

ID: 212648257

Teacher: Anatoly Peymer

Alternative: תכנון ותכנות מערכות

Submission date: 27.5.2020



תוכן עניינים:

[מבוא 3](#_Toc41407360)

[תכולת הספר 3](#_Toc41407361)

[רקע לפרויקט 3](#_Toc41407362)

[תהליך המחקר 3](#_Toc41407363)

[אתגרים מרכזיים 4](#_Toc41407364)

[בעיות שהיו לי 4](#_Toc41407365)

[למה בחרתי את הפרויקט הזה 4](#_Toc41407366)

[המוטיבציה לבניית הפרויקט 4](#_Toc41407367)

[מבנה הפרויקט 5](#_Toc41407368)

[הצגת הפתרון המוצע 5](#_Toc41407369)

[מבט על כל מחלקה 5](#_Toc41407370)

[Top Down Level Design 7](#_Toc41407371)

[זרימת המידע בין היחידות השונות 8](#_Toc41407372)

[ארכיטקטורת רשת 11](#_Toc41407373)

[Server UMLתרשים 12](#_Toc41407374)

[Game UMLתרשים 14](#_Toc41407375)

[Admin UMLתרשים 16](#_Toc41407376)

[באלו טכנולוגיות נעשה שימוש בפרויקט 17](#_Toc41407377)

[מדריך למשתמש 17](#_Toc41407378)

[המדריך למשתמש במשחק 17](#_Toc41407379)

[חוקי המשחק 18](#_Toc41407380)

[היררכית מסכים 18](#_Toc41407381)

[הסברים על חלונות 20](#_Toc41407382)

[מאגר נתונים 27](#_Toc41407383)

[הסבר על הקבצים. 27](#_Toc41407384)

[הסבר מסוכם 27](#_Toc41407385)

[מדריך למפתח 29](#_Toc41407386)

[קבצי השרת 29](#_Toc41407387)

[קבצי המשחק 52](#_Toc41407388)

[קבצי מערכת המנהלים 75](#_Toc41407389)

[רפלקציה 82](#_Toc41407390)

[ביבליוגרפיה 83](#_Toc41407391)

[אתרים שהשתמשתי בהם 83](#_Toc41407392)

# מבוא

## תכולת הספר

הספר מכיל תשובות לשאלות כמו למה ואיך בחרתי בפרויקט הזה, מה היתרונות שלו ואיך בדיוק בניתי אותו, בתוך הספר אפשר למצוא את הארכיטקטורה של הקוד, רוב הפונקציות והמחלקות שהשתמשתי בכדי ליצור את הפרויקט והכילים שעזרו לי לבנות אותו.

בנוסף לכך הספר מכיל מדריך למשתמש בכדי שהמשתמש ידע בדיוק איך לשחק במשחק. בסוף הספר ישנה רפלקציה שבאה לראות לקורא של הספר את המחשבות שלי על הפרויקט ואת התהליך הארוך שעברתי בכדי להוציא לפועל את הפרויקט, בסוף ממש אפשר לראות את האתרים שבהם השתמשתי בכדי לפתור בעיות..

## רקע לפרויקט

בתחילת שנה כשביקשו ממנו לבחור נושא לפרויקט חשבתי על המון אופציות, רובם בכלל לא היו משחקי מחשב, אך לאחר חשיבה עמוקה חשבתי על משחק אסטרטגיה, משחק שבו המטרה המרכזית היא להבין בתוך זמן קצר על ידי פגיעות היריב איך הראש שלו עובד.. ובעזרת זה למצוא את כל הספינות )C#)שהוא מיקם ולפוצץ אותם. את הפרויקט כולו כתבתי בשפת התכנות שאני הכי מתמצא בה

תחילה בניתי את בסיס התקשורת, בניתי שרת ולקוח שעובדים על ידי מחלקות שלמות כך שתקשורת הקליינט תהיה בתוך מחלקה אחת שמקבלת הודעות מהשרת ומעבירה לקליינט ובנוסף לכך מעבירה הודעות מהקליינט לשרת בדרך ההפוכה, בעוד שהשרת פותח אובייקט במחלקת התקשורת שדואגת על ידי פרוטוקול ספציפי לשלוח לקליינט בדיוק מה שהוא ביקש.

למשך תקופה ארוכה כתבתי את המשחק והשרת, בעוד שבראש היה לי עוד רעיון לסקריפט נוסף להוסיף לפרויקט, כשראיתי שיש לי את הזמן לעשות זאת, העברתי את כל השימוש שלי במערכת הנתונים החיצונית אס-קיו-אל לייט למערכת שבעזרת קבצי טקסט מחזיקה קבצי מידע איכותיים בעלי יכולות עריכה נוחות בהרבה יותר מהמערכת של אס-קיו-אל לייט, בנוסף לכך והכי חשוב, למערכת זו יש את האופציה לעבוד גם מחוץ למחשב שמריץ את השרת ובכך מאפשרת אפילו לכמה אנשים שאפשר לסמוך עליהם ובצע שינויים קריטים ומשמעותיים בקבצי המידע, כמובן שבכדי להיכנס למערכת זו ישנה הגנה שמחייבת את הקליינט להזין נתונים מאומתים של אדם בעל גישה למערכת בכדי שיוכל להיחשף ולשנות נתונים בקבצי המידע

## תהליך המחקר

במהלך כתיבת הפרויקט למדתי המון דברים בעלי ערך לכתיבת קוד, בערך כשמונים אחוזים ממה שכתבתי בפרויקט למדתי במהלך כתיבת הפרויקט.. לדוגמא, כל העיצוב של המשחק ב דאבליו-פי-אף למדתי שעות, מהחלק הכי קטן של לחלק את המסך עד להוספת והורדת אובייקטים מין המסך.. בנוסף לקח למדתי לעבוד עם קבצים.. דבר שרציתי ללמוד מאוד במהלך הפרויקט וכיום יש לי מערכת ניהול ושינוי קבצים חיצונית למחשב שבו ממוקמים הקבצים. בנוגע לפופולאריות של המשחק, המשחק מוכר כיום כמשחק קופסא בעיקר, ורוב משחקי המחשב שלו חסרי עיצוב גרפי יפה, בנוסף למראה המשחק שלי שומר את כמות הניצחונות וההפסדים שלך ומאפשר לך מהביע דעה בפני מנהלי המשחק דבר הגורם לחיבור רגשי של המשתמשים בפרויקט למשחק, המשחק שלי בנוסף למשחק עצמו מהווה גם סביבת משחק שאינה שטחית, ובכך מאפשרת לאנשים לשחק במשחק לאורך זמן מבלי להשתעמם ממנו ואפילו להתמכר אליו.

## אתגרים מרכזיים

### בעיות שהיו לי

במהלך בניית הפרויקט אני נתקלתי בהמון בעיות, הרבה יותר משאוכל לתאר. האתגר הראשון שלי היה לבחור את נושא הפרויקט, תחילה חשבתי רבות על תוכנות מסוימות אך מכוון שרציתי לבחור פרויקט שדרכו אלמד גם על דאבליו- פי-אף וגם על התעסקות בקבצים בחרתי בפרויקט הזה, לאחר חודשיים של תכנון הפרויקט התחלתי לבנות את השרת והלקוח, דבר זה היה לא פשוט בכלל, למדתי המון שיטות של העברת מסרים ותקשורת אך בכל זאת דרכים אלו היו לא מתאימות או מסובכות מידי, לאחר התייעצות עם עמיתי לכיתה החלטתי ליצור שרת ולקוח שמתקשרים דרך מחלקות מסוימות שאחראיות על כל התקשורת, כך שבפועל במהלך כתיבת המשחק עצמו כל התקשורת נעשתה על ידי שתי פקודות בלבד שדרכם הייתי שולח ומקבל, לאחר שפתרתי בעיה זו התחלתי לעבוד על בניית מסד הנתונים, דבר זה גבה ממני חשיבה רבה מכוון שהייתי צריך לתכנן איך אני רוצה שהמידע ישמר אצלי בשרת, לבסוף לאחר חשיבה רבה הצלחתי לחשוב על פורמט מצויין לשמירת נתוני משתמש; שורה ראשונה תכיל את שם המשתמש, שורה שנייה תכיל את הסיסמא שלו, ושורה שלישית תכיל את נתוני המשחק שלו בצורה הבאה (כמות הפסדים/כמות ניצחונות/רמת אדמין) כל דבר הזה מתרכז לתוך קובץ טסט אחד המכיל את כל הנתונים, הסיבה שבחרתי לעשות את מאגר המידע בצורה כזאת ולא אס-קיו-אל לייט היא שלא רציתי להשתמש בשום תוכנה שלא בניתי, העדפתי להראות שיש לי גם את היכולת לבנות מאגר מידע שיש לי שליטה בכל מה שקורה בתוכו גם אם אני לא על המחשב שמריץ אותו, ובכך להוסיף עוד משהו לפרויקט. אך זה לא הבעיה היחידה שהייתה לי, כשהתחלתי לבנות את המשחק חשבתי על תכנון המסך, איפה אני ימקם כל דבר כך שיהיה ברור לעין ונוח למשחק, לאחר חשיבה רבה החלטתי על צורה מסוימת ובניתי אותה, לאחר חלק זה התחלתי עם מיקום הספינות, ובכדי למקם ספינות תחילה צריך ליצור אותם בתור תמונה, הבעיה הייתה כשחיפשתי המון תמונה שתתאים אבל לא מצאתי הרבה, חיפושים אלה ערכו כמה ימים, עברתי על כל תמונה רלוונטית אך רובם לא באמת היו אופציות, לאחר שראיתי שאין לי הרבה אופציות החלטתי לקחת צעד קצת יותר אינטנסיבי ולקחת תמונה שקרובה למה שאני מחפש ולערוך אותה.

### למה בחרתי את הפרויקט הזה

בחרתי את המשחק הזה משלשה סיבות

1. חיפשתי פרויקט שבו אוכל לעבוד על נושאים מסוימים כמו בניית מסד נתונים ועיצוב חלונות

2. אני בכללי אוהב משחקי אסטרטגיה וחשיבה כמו המשחק הזה

3. לאחר שהחלטתי שאני יעשה משחק חשבתי על המון משחקים, משחק זה בלט מעל כולם מכוון שהוא מאוד קל להבנה ותפעול.

### המוטיבציה לבניית הפרויקט

כשהתחלתי לבנות את הפרויקט לא ידעתי הרבה על תכנות, וגם לא הרבה על מה שעשיתי בפועל, רוב הפרויקט שלי היה למידה תוך כדי תנועה, משמע כתבתי את הפרויקט תוך כדי שאני לומד את מה שאני עושה, ובכך צברתי ניסיון בניית פרויקטים בסדר גודל כזה. המוטיבציה העיקרית שלי הייתה כמו של כל מתכנת, כשאתה כותב משהו ומשקיע אתה רואה את זה עובד בסופו של דבר זה נותן לך מוטיבציה להמשיך, דבר זה משך אותי קדימה עד שכשסיימתי את המשחק עצמו והתחלתי לבנות את מערכת המנהלים לא יכולתי להפסיק ואפילו שתכננתי לעשות פחות ממה שעשיתי בסוף הוספתי עוד פיצ'רים לפרויקט.

# מבנה הפרויקט

## הצגת הפתרון המוצע

כיום המשחק צוללות הינו משחק מאוד מוכר שהרבה אנשים מכירים ומשחקים בו, אך רוב האנשים מכירים את המשחק הזה כמשחק קופסא או במקרה הכי טוב כמה ריבועים באינטרנט, הפרויקט שלי מציע רמה חדשה של המשחק, כשהמשחק יותר וויזואלי נוח ומחובר יותר למשתמשים

## מבט על כל מחלקה

(חלק זה כתוב גם במדריך למתכנת)

**Server classes:**

ClientHandler.cs

מחלקה זו אחראית על טיפול בלקוח, משמע השרת יוצר לכל לקוח טיפוס מין מחלקה זו וכל טיפוס כזה אחראי על העברת נתונים בין השרת ללקוח שעליו הוא ממונה, הקלט של המחלקה הוא מטיפוס מחרוזת שמועברת מהלקוח לחלק זה של השרת, השרת מעבד את הבקשה משנה דברים בקבצים או בנתוני במשחק אם צריך ומחזיר מחרוזת

Game.cs

מחלקה זו אחראית על משחק עצמו, החזקת האיידי של השחקנים ובניית המגרש, אחרי שהמחלקה בונה את המגרש והמשחק עובר לפגיעות, המשחק עובד באמצעות פונקציה מרכזית במחלקה שאחראית על בדיקת המיקום של הפגיעות ושליחה חזרה למחלקה המרכזית של השרת

program.cs

מחלקה זו אחראית על הרצת התוכנה ובדיקת תקינות קבצי המידע שבה (כל קבצי הטקסט) כך ששאר המחלקות יוכלו לרוץ באופן חלק, בנוסף לכך מחלקה זו כותבת לקובץ לוג כל פעם שהיא רצה.

User.cs

מחלקה זו אחראית על בניית משתמש לכל שחקן שמתחבר כך שהשרת יכול לשלוף את זה בכל רגע שיצטרך, בנוסף לכך הפונקציה אחראית על שינויים בתוך קובץ

בו היא משנה כמות ניצחונות הפסדים ורמת אדמין..Database.txt

Server.cs

מחלקה זו אחראית על בניית השרת עצמו והוספת לקוחות עליו (מטיפוס קליינט הנדלר).

**Game classes:**

MainWindow.xaml.cs

מחלקה זו אחראית על טיפול במסך הלקוח, משמע כל דבר שצריך להציג או לקבל על ידי לחיצה או הזנה של נתונים שלא כוללת את המשחק עצמו קורה במחלקה הזאת

ServerHandler.cs

מחלקה זו אחראית העברת מסרים בין הלקוח לשרת, מחלקה זו היא בעצם מקשרת בין השרת ללקוח בצורה של העברה וקבלה של מחרוזות

Game.xaml.cs

מחלקה זו אחראית על תצוגת המשחק עצמו, בעזרת הפונקציה המקשרת בין הלקוח לשרת הפונקציה הזו מקבלת ומעבירה מידע ובכך יודעת מה להציג על המסך בזמן המשחק.

**Admin classes:**

MainWindow.xaml.cs

מחלקה זו אחראית על טיפול במסך הלקוח בשלב הכניסה והעברת נתוני הכניסה למחלקה שאחראית על התקשורת בין השרת ללקוח

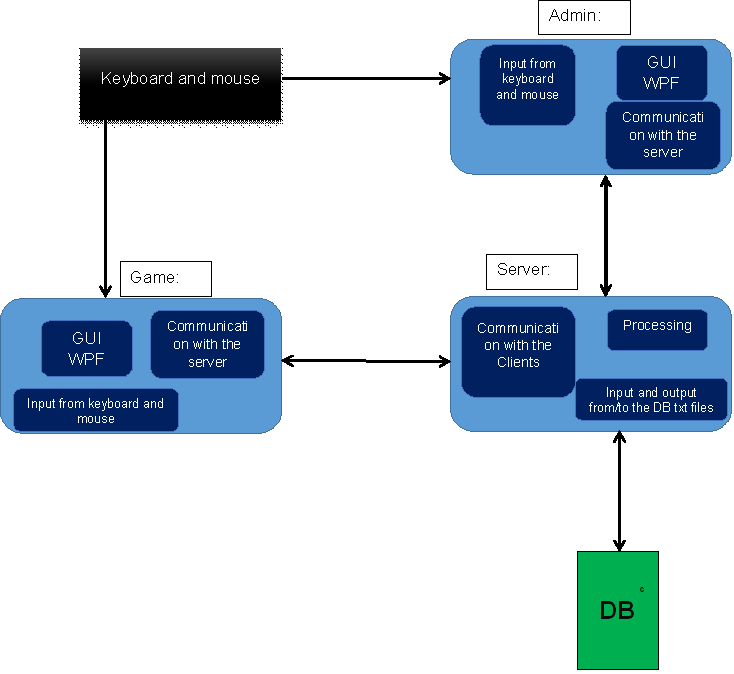
MainAdminWindow.xaml.cs

מחלקה זו אחראית על טיפול במסך הלקוח כאשר הלקוח מאומת כאדמין ומבצע פעולות אדמין כמו צפייה בצאט או שינוי סיסמא למשתמש, הקלט של מחלקה זו היא הכפתורים והנתונים שהלקוח מזין

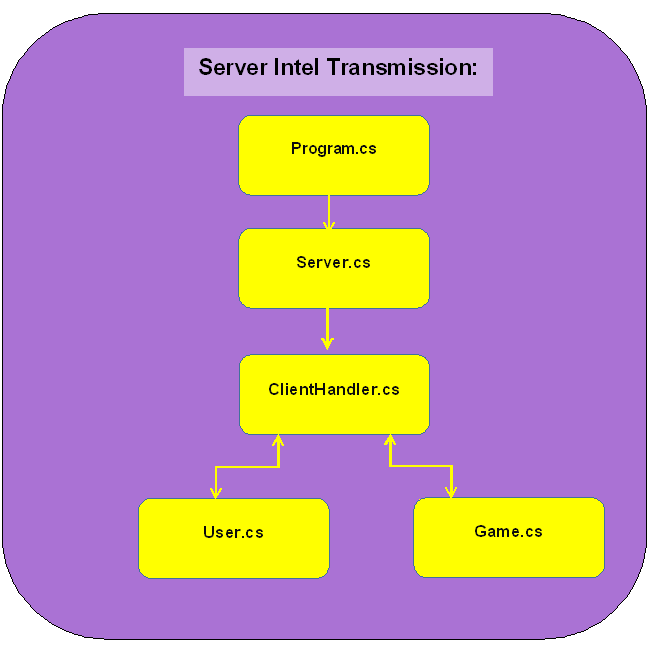
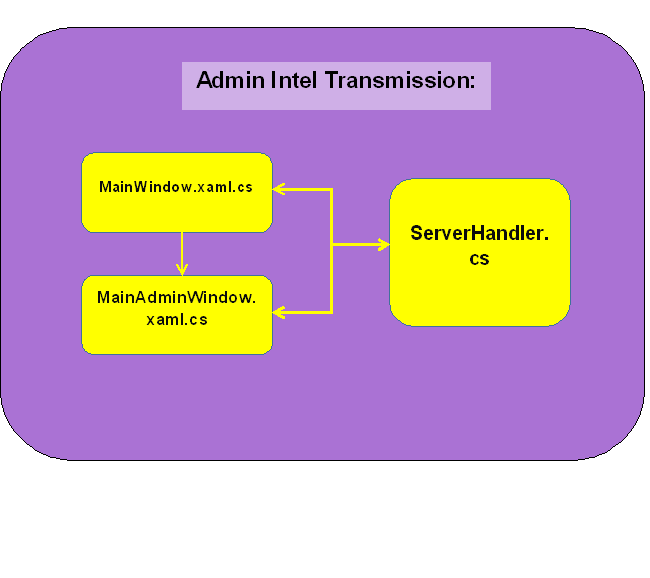
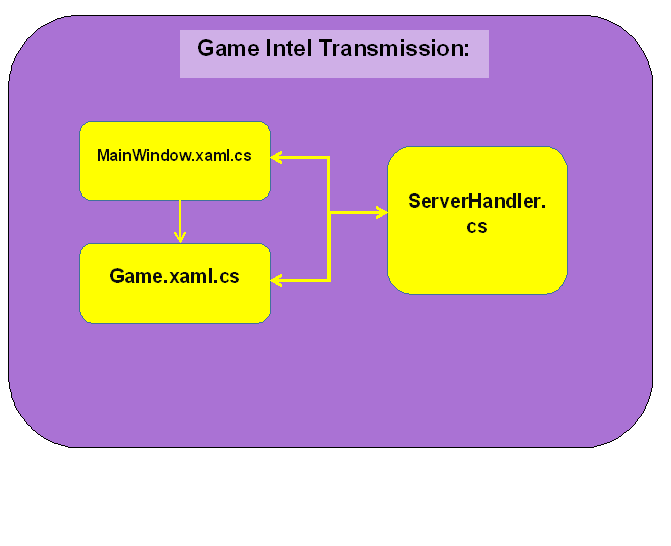
ServerHandler.cs

מחלקה זו אחראית העברת מסרים בין הלקוח לשרת, מחלקה זו היא בעצם מקשרת בין השרת ללקוח בצורה של העברה וקבלה של מחרוזות

## Top Down Level Design



## זרימת המידע בין היחידות השונות



## ארכיטקטורת רשת

הפרויקט משתמש בשיטת שרת לקוח ועובד בשיטת סטייט-פול. משמע זוכר את השלבים שהלקוח נמצא בו, השרת עובד בלי הצפנה אך כן יש בו הגנה.

השרת עובד בשיטה של פרוטוקול שבו נמצאים קודים מובנים, השרת מזהה את הקוד ובכך יודע איך להתייחס לכל בקשה ולהחזיר או לשנות לפי הבקשה. בנוגע להגנה, ההגנה נעשת על ידי תנאים, דוגמא רק שחקן שהתחבר למערכת על ידי מתן שם משתמש וסיסמא או שנרשם יכול לבצע קודים ששייכים לשחקנים, ובנוסף לכך רק משתמש שהתחבר למערכת המנהלים תוך כדי מתן שם משתמש וסיסמא של משתמש המוכר כאדימין יכול לבצע ולתת קודים ששייכים לאדמינים ולקבל את הנתונים שהוא ביקש.

פרוטוקול השרת עובד לפי הקודים הבאים:

LOG – אחראי על כניסת משתמשים, מקבל מחרוזת המורכבת משם המשתמש והסיסמא של המשתמש ובודקת אם הוא קיים במערכת, אם כן מכניסה אותו ומחזירה הודעה ללקוח

REG – אחראי על הרשמת משתמשים, מקבל מחרוזת המורכבת משם המשתמש והסיסמא של המשתמש ובודק אם אפשרי לרשום אותו, אם כן מחזיר תשובה חיובית ואם לא תשובה שלילית

LAG – אחראי על כניסת מנהלים, מקבל מחרוזת המורכבת משם המשתמש והסיסמא של המשתמש ובודקת אם הוא קיים במערכת והאם הוא אדמין, אם כן מכניסה אותו ומחזירה הודעה ללקוח

OUT – מוציא את המשתמש ממצב של העברת הודעות ובכך מונע עבודה מיותרת של השרת

כל אלה נהל הם קודים שאפשריים לכל לקוח שנכנס לשרת, בין אם הוא רשום או לא.

הקודים הבאים הינם אפשריים רק למנהלים שאושרו על ידי השרת.

AFP – "אדמין חפש שחקן" קוד זה אחראי על חיפוש משתמש ספציפי במערכת. אם השחקן באמת קיים במערכת המערכת תשלח את נתוניו למנהל, ואם הוזן מידע לא נכון תציין את זה ותשלח.

SET – קוד זה אחראי על הפעלת פונקציה שמשנה בגדול את נתוני שחקן, שם השחקן והנתונים שלו מוזנים ישירות לפונקציה בצורת מחרוזת

GDB – שולח את כל המסד נתונים שמחזיק את נתוני משתמשים אל האדמין

EDB – עורך את המסד נתונים על ידי קבלת מסד נתונים חדש

GLG – שולח את כל קובץ לוג אל האדמין

GCC – שולח את כל קובץ צאט אל האדמין, אם הקובץ ריק, שולח הודעה שהקובץ ריק

DCC – אחראי לריקון כל קובץ צאט

הקודים הבאים תקפים רק על מי שאושר כשחקן

CHT – הוספת הודעות לקובץ צאט.

PLY – כששחקן מנסה להתחבר למשחק הוא מפעיל את הקוד הזה, הקוד הזה בודק אם יש עוד שחקן שמחכה ואם לא שולח הודעה לחכות, ברגע ששתיי שחקנים מפעילים את קוד הזה המונה מתאפס ומפעיל משחק בין שתי השחקנים שמחכים.

STS – שולח את נתוני המשתמש אל המשתמש על פי שם המשתמש והסיסמא שהוא התחבר איתם

GEN – קבל שם יריב, משמע הלקוח שולח בקשה לקבל את שם היריב כך שיוכל להציג אותו

SPP – קוד זה נועד להודיע ליריב שהמשתמש ששלח אותו סיים להכונן, ובכך השרת ידע להתחיל את המשחק מתי שצריך

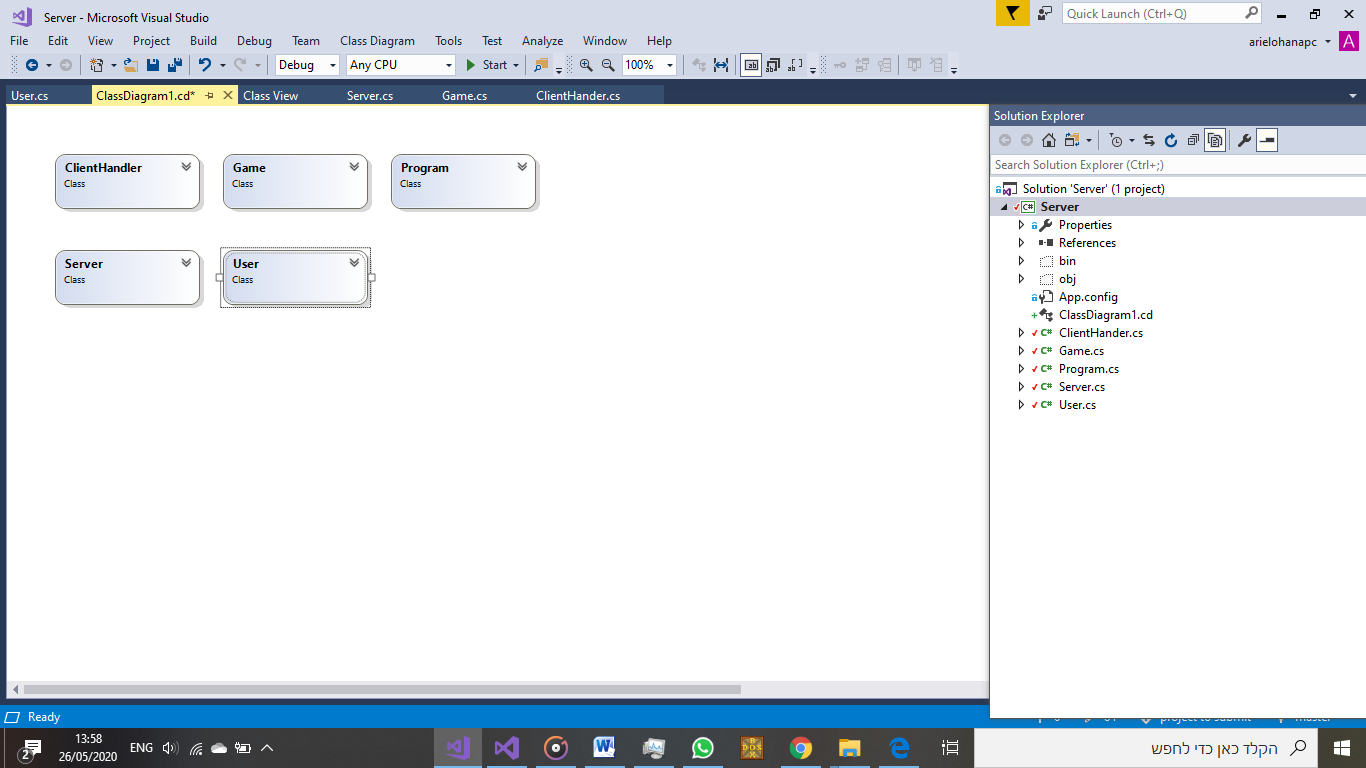
FLD – כל שחקן יוצר בחצי שלו את המגרש, ולאחר שמסיים שולח לשרת את המגרש שלו. קוד זה נועד בכדי לדעת לקבל את המגרש שהזין הלקוח

HIT – פגיעה בשחקן יריב, קוד זה נועד בכדי לקבל נקודה ספציפית מהלקוח ולבדוק אם פגע

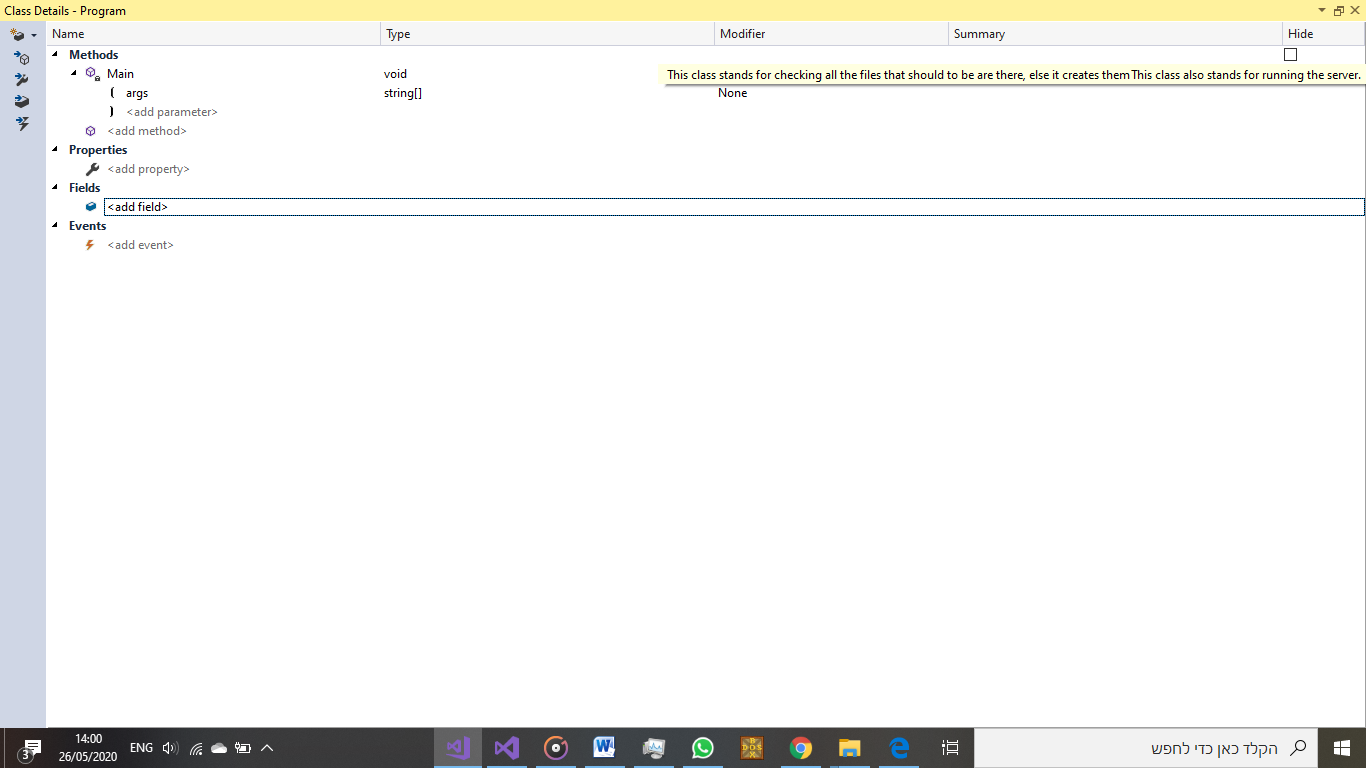
WIN – קוד זה נועד בכדי לקבל הודעת ניצחון על ידי אחד מין השחקנים, הודעה זו גם מגיעה לשחקן המפסיד.

CHW – קוד זה בודק אם השחקן השני ניצח לפני שמעבירים את התור.

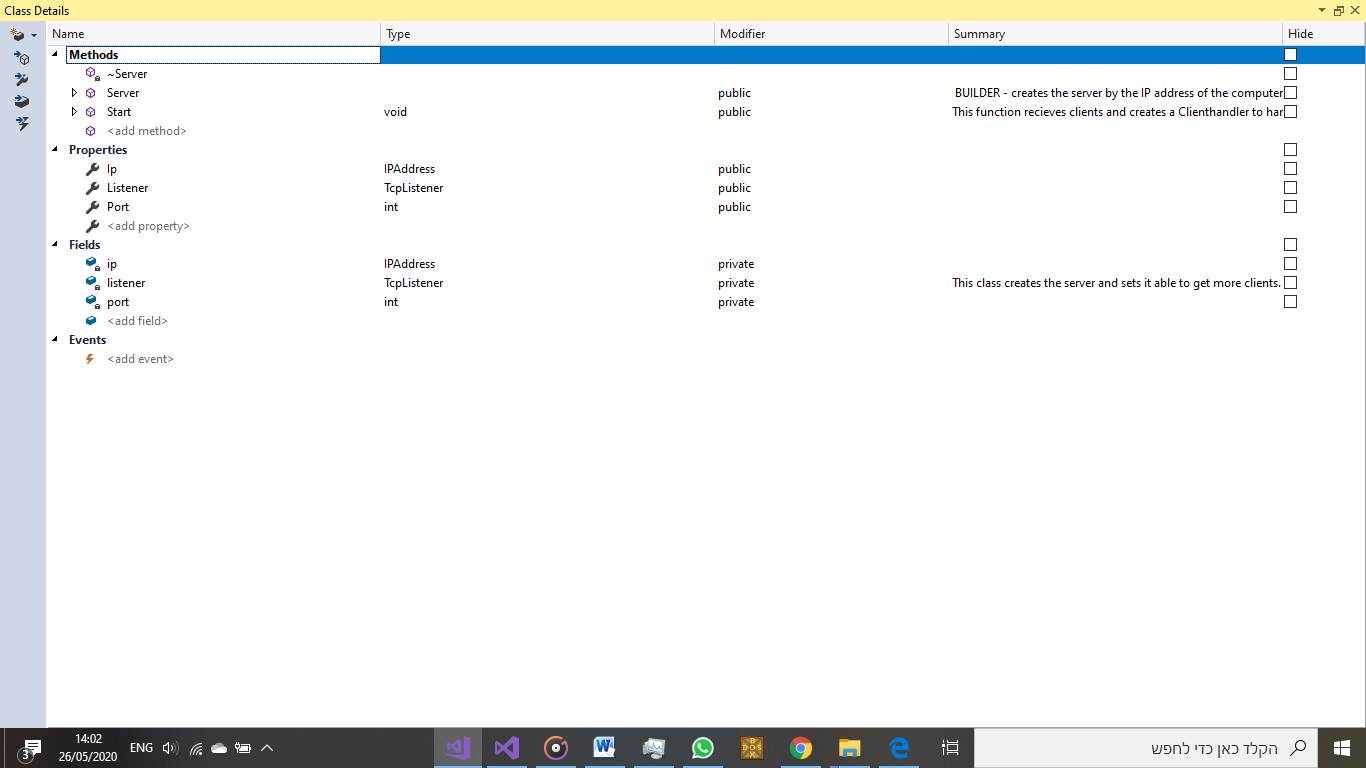
## Server UMLתרשים



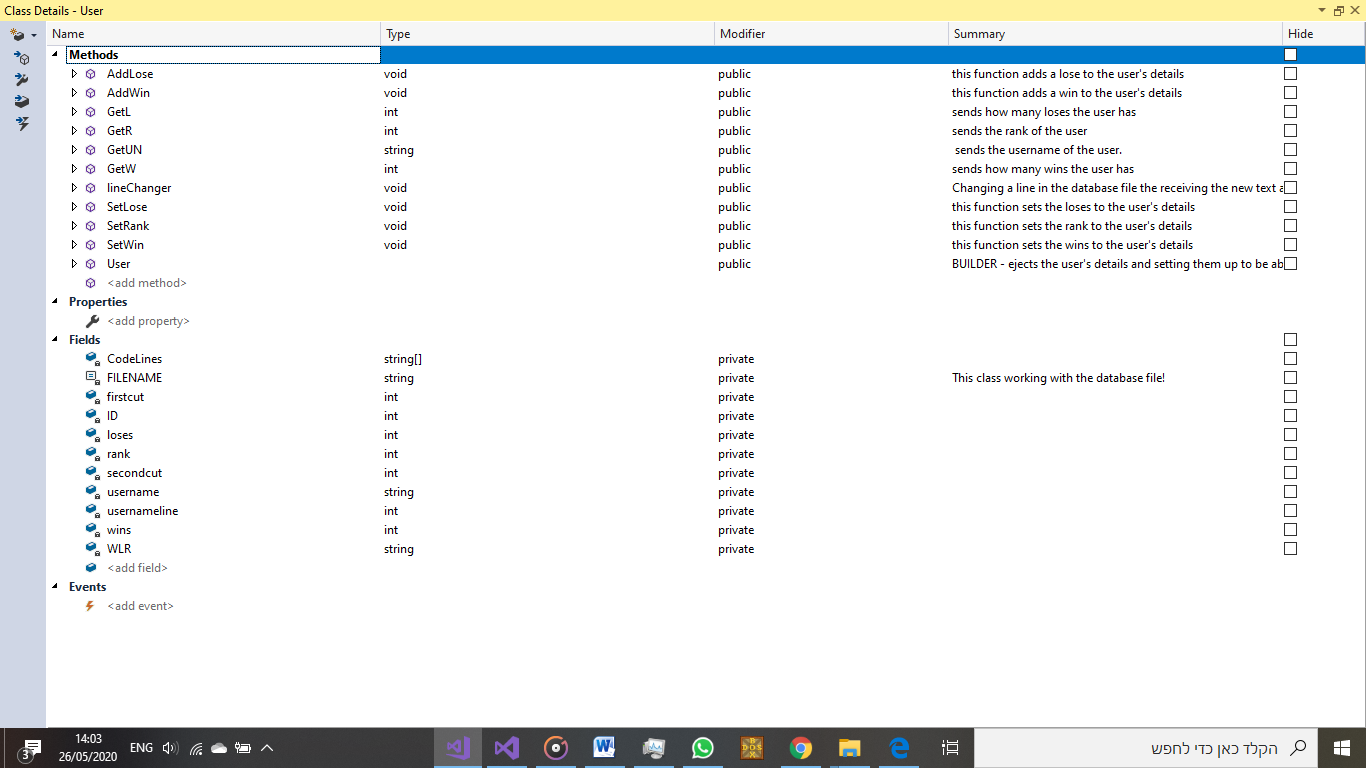
Program:



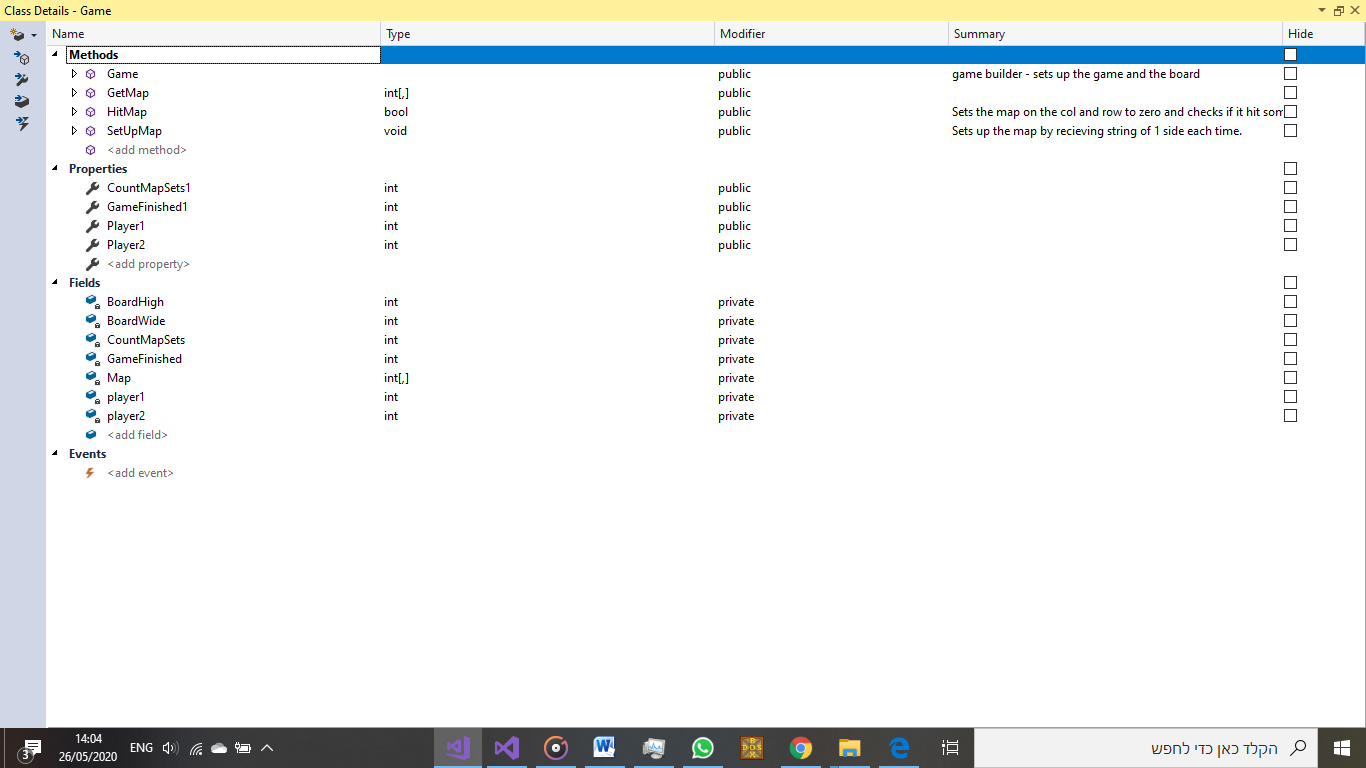
Server:



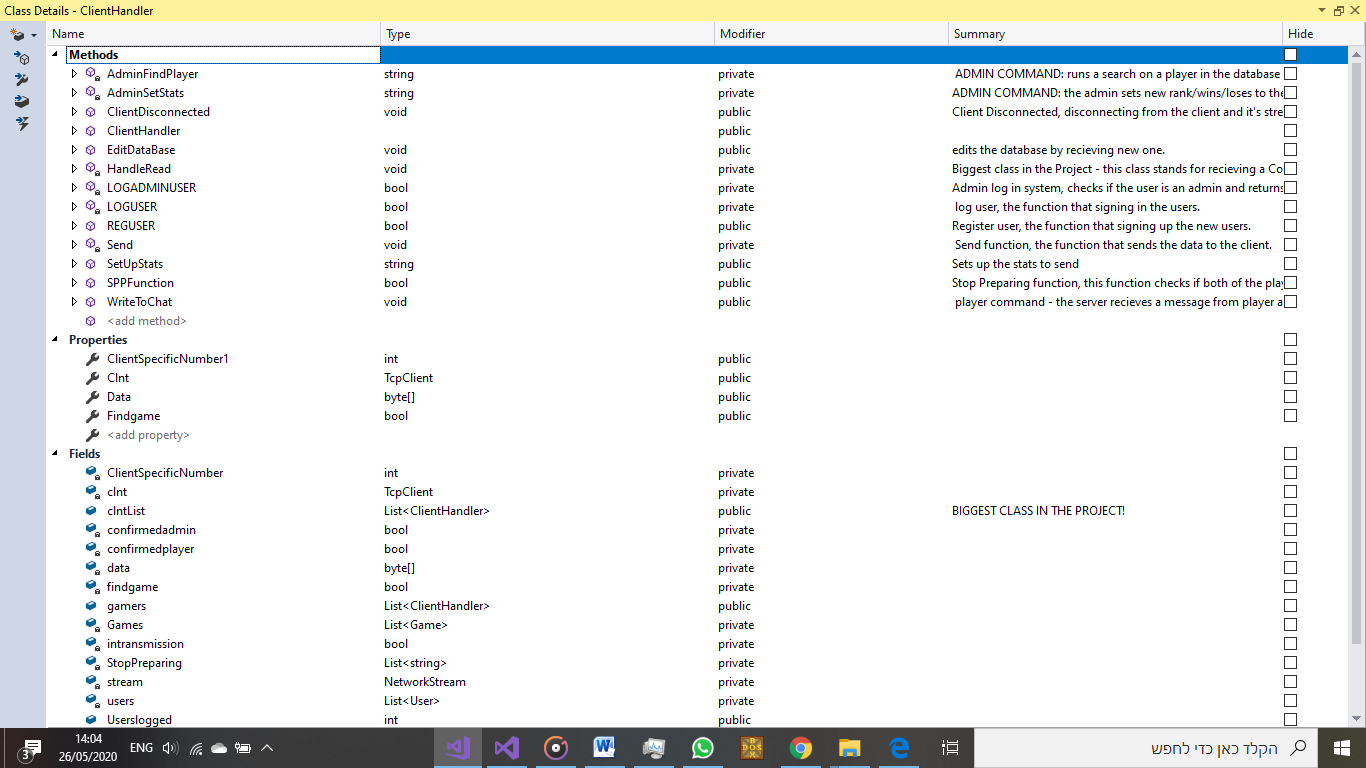
User:



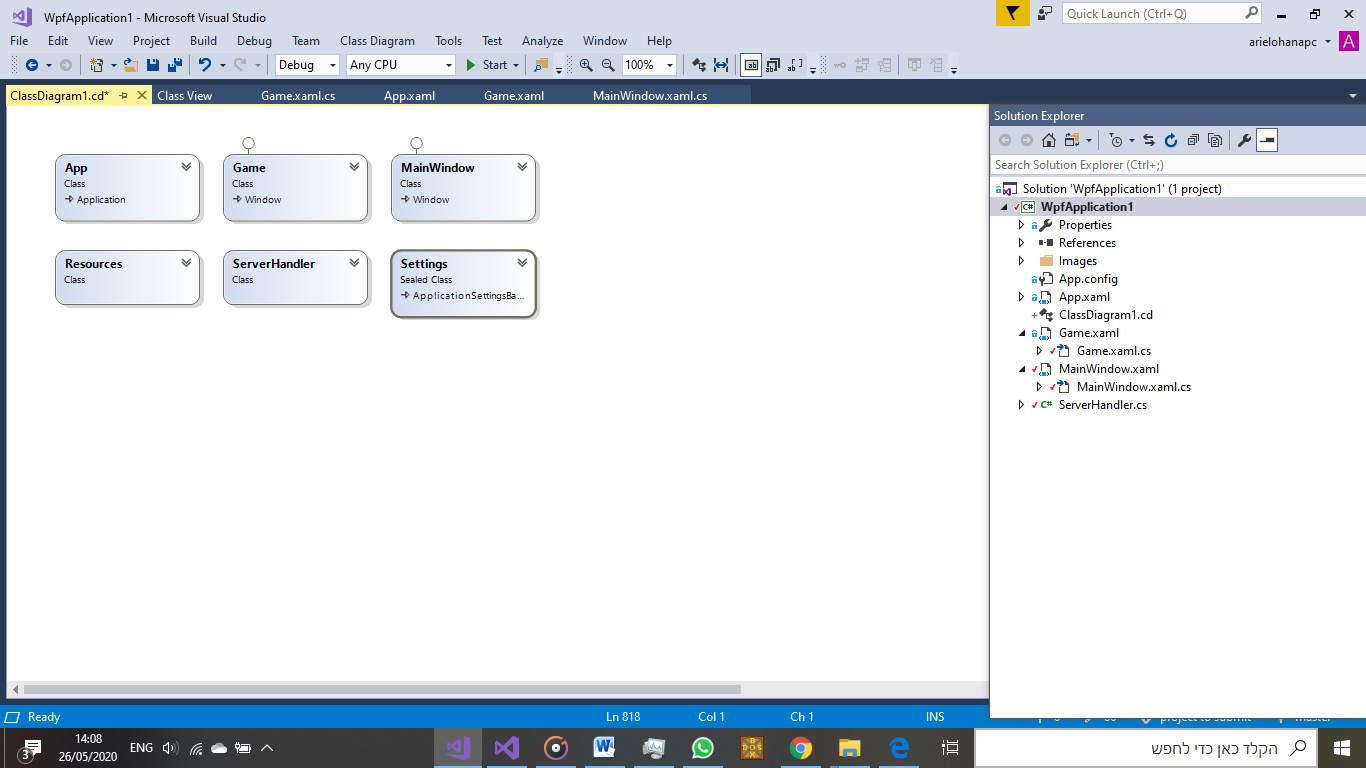
Game:



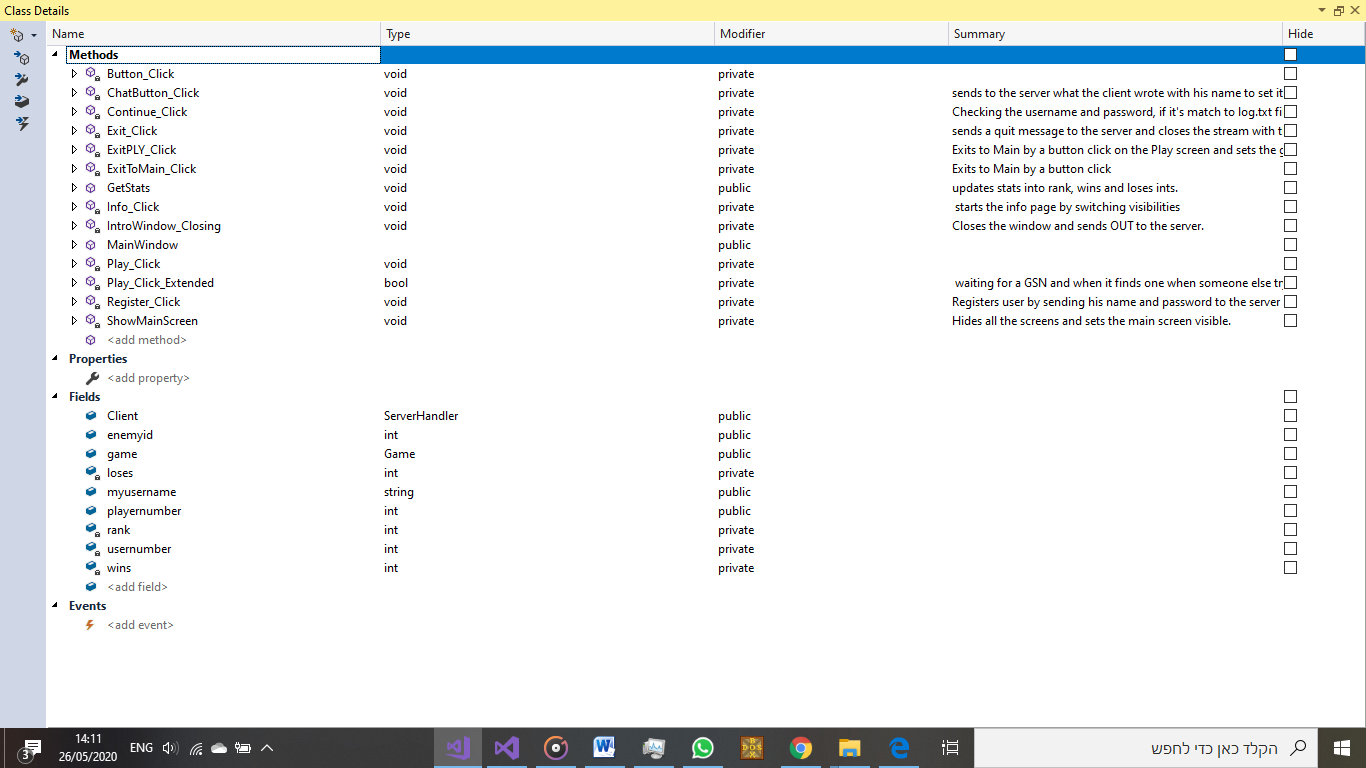
ClientHandler:



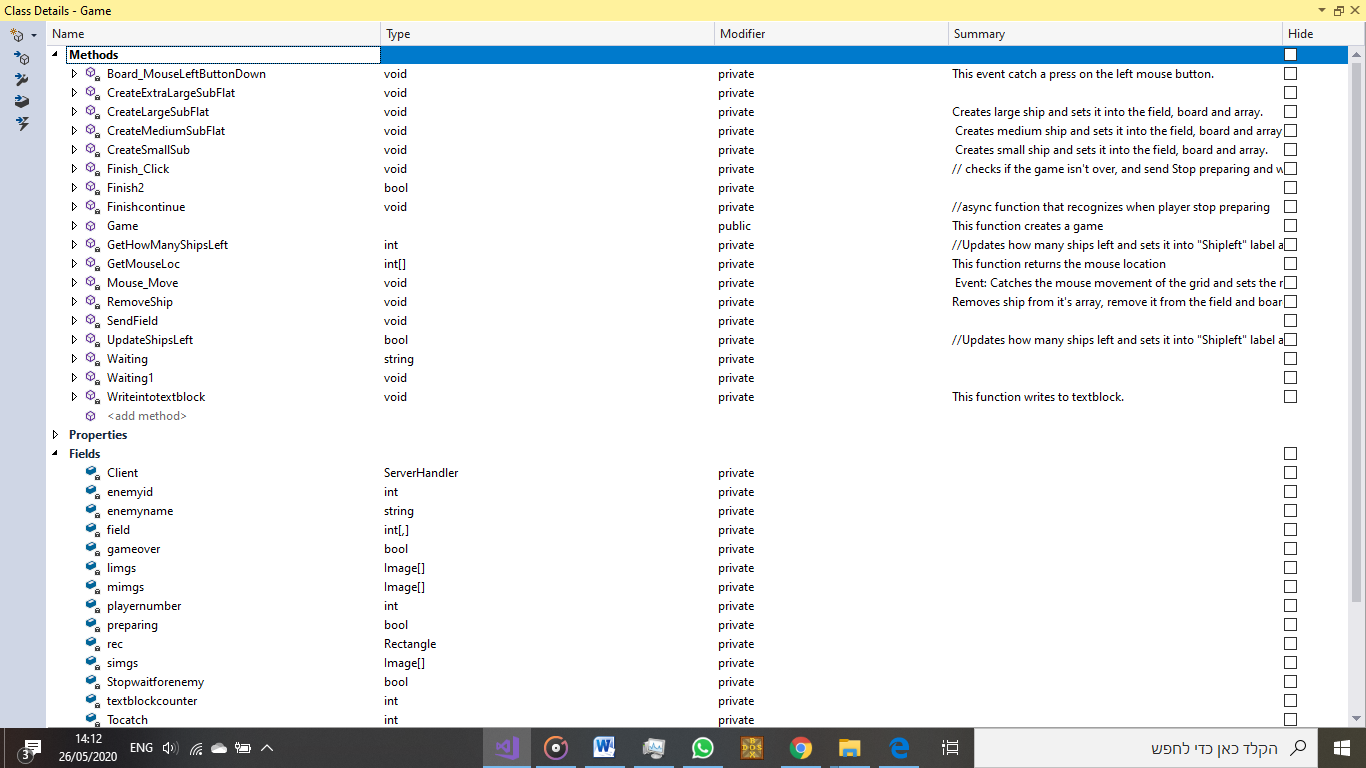
## Game UMLתרשים

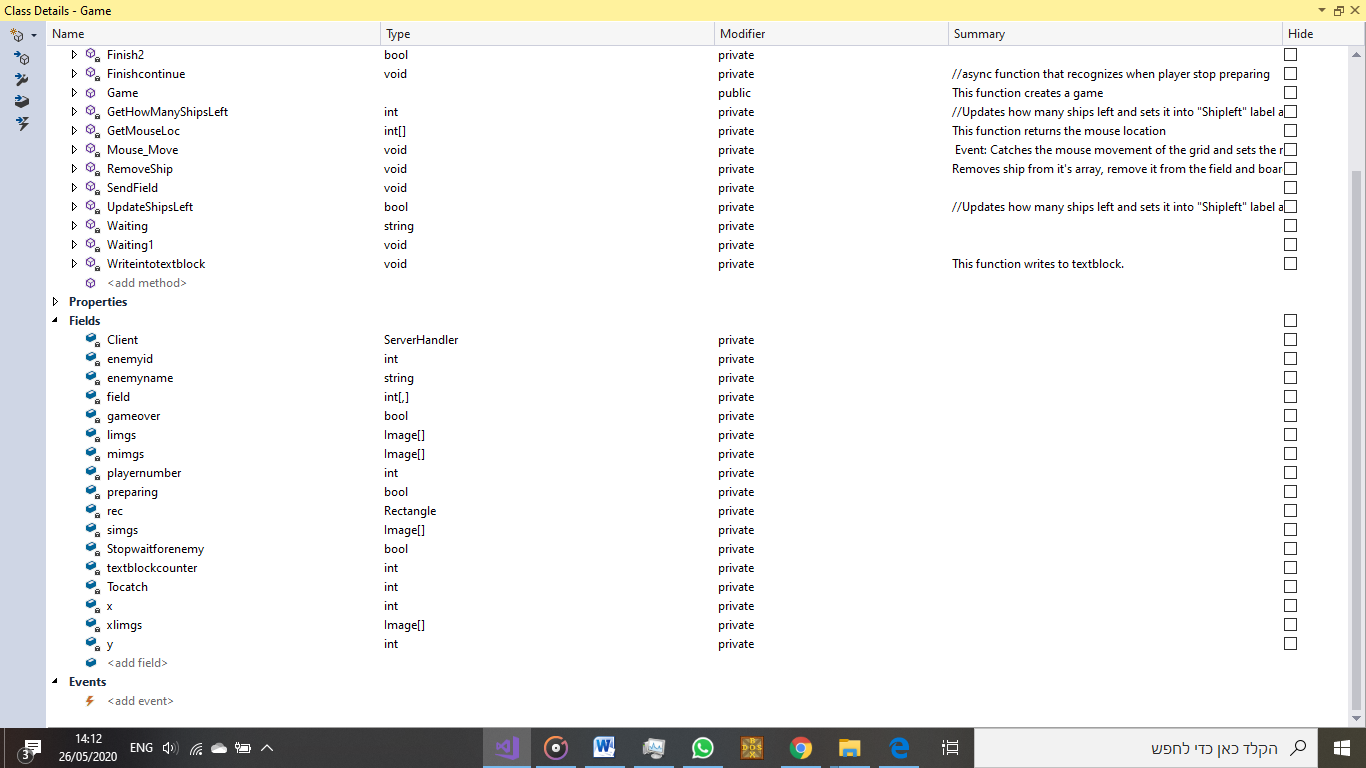


MainWindow:

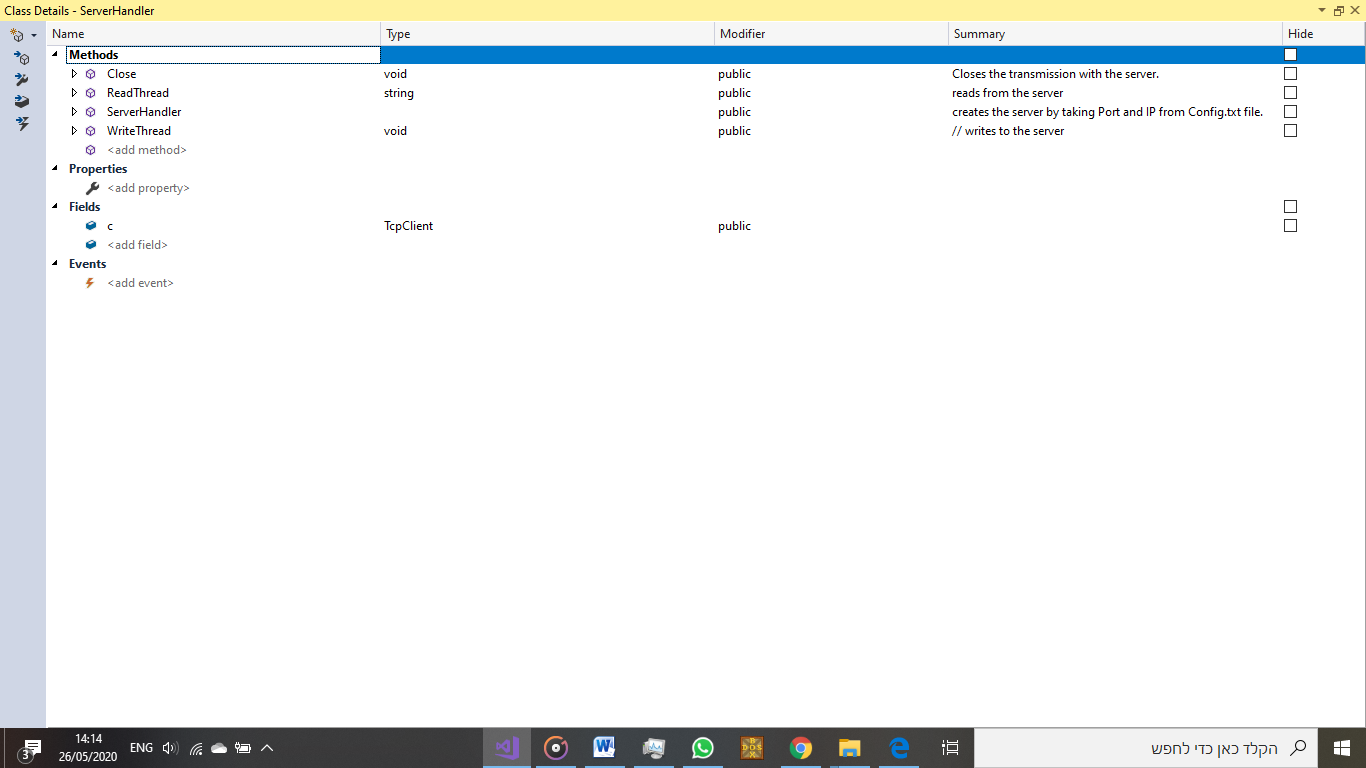


Game:

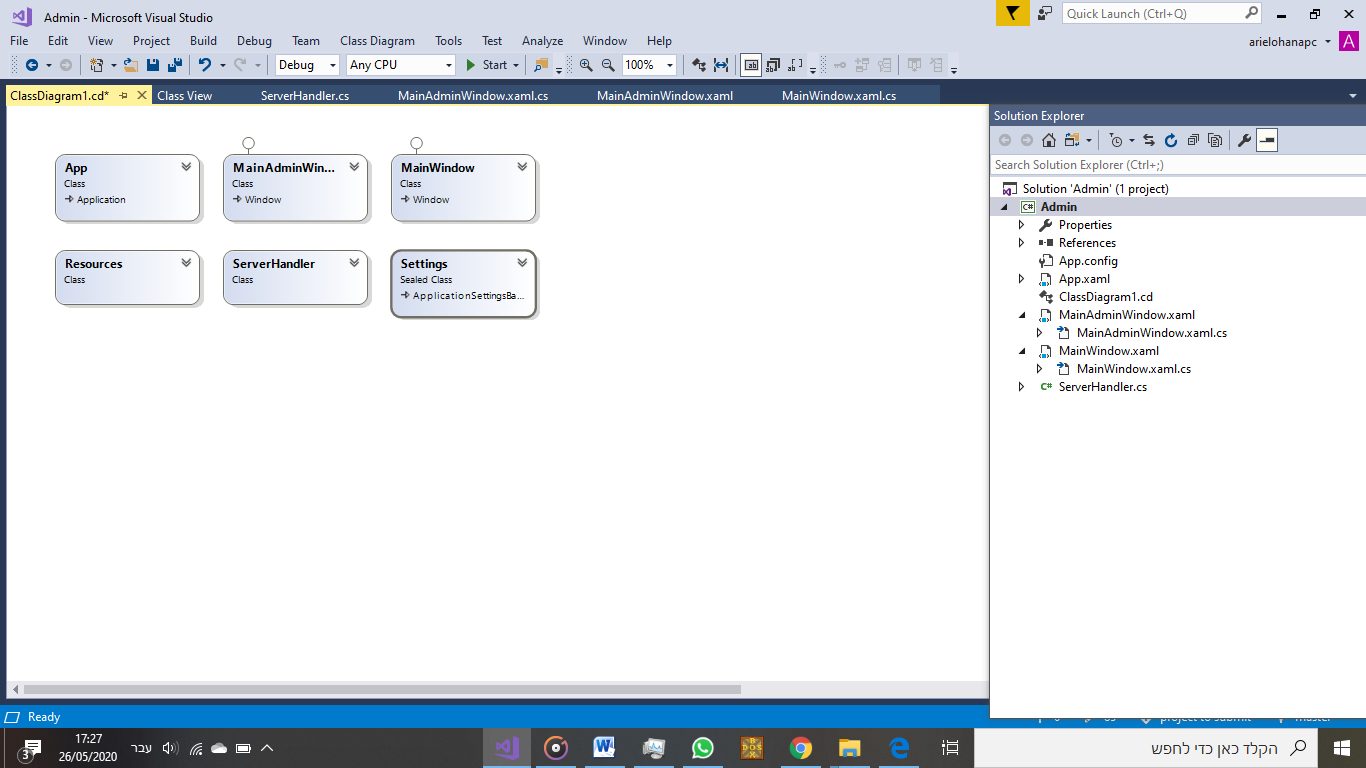




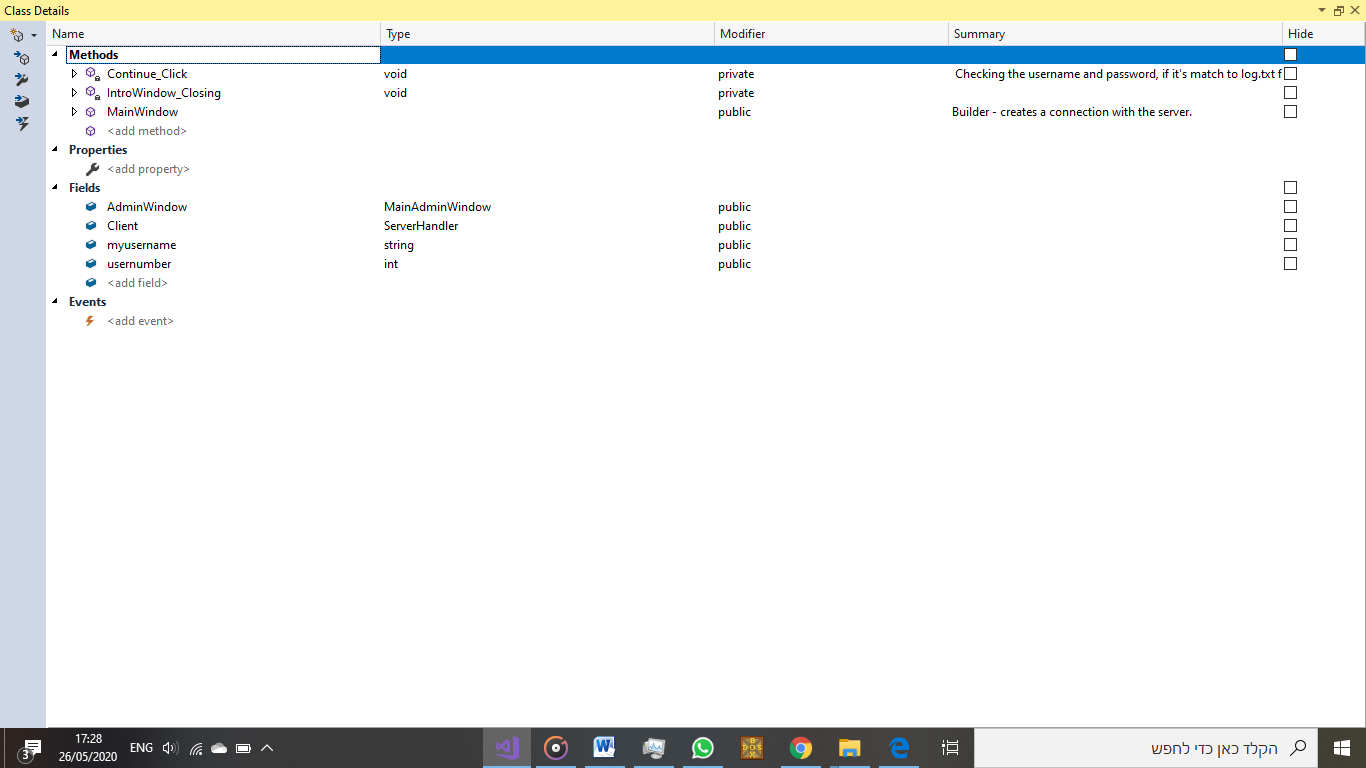
ServerHandler:



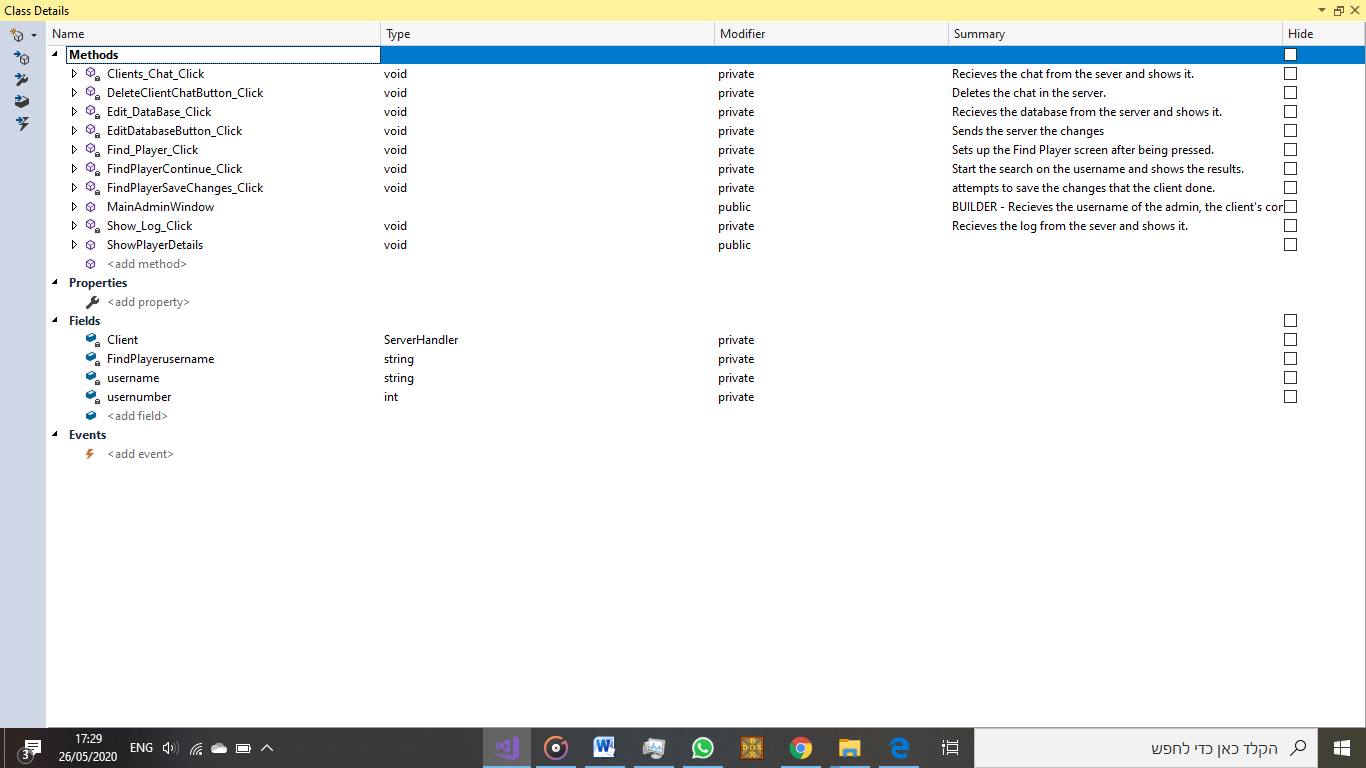
## Admin UMLתרשים



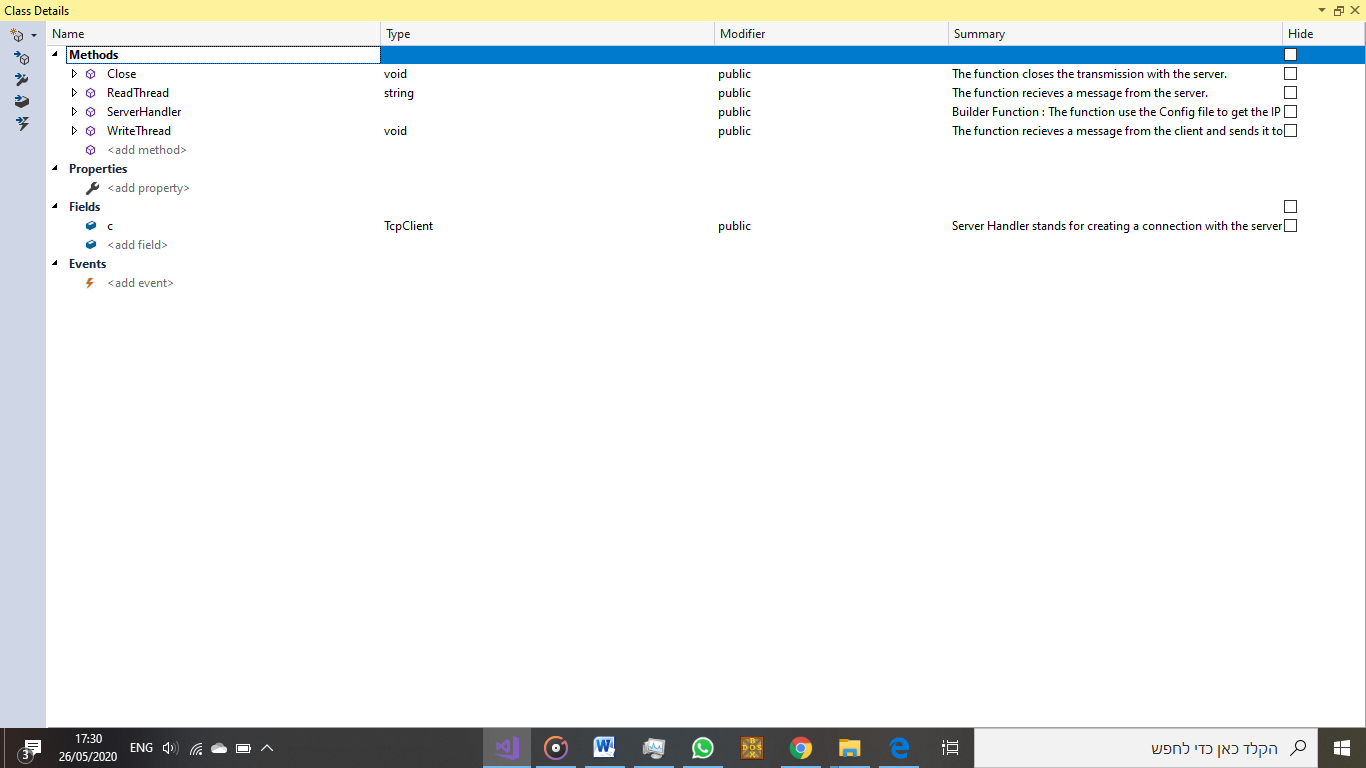
MainWindow:



MainAdminWindow:



ServerHandler:



## באלו טכנולוגיות נעשה שימוש בפרויקט

בפרויקט לא נעשה שימוש בשום טכנולוגיות חיצוניות מלבד וויזואל סטודיו וקבצי טקסט

# מדריך למשתמש

## המדריך למשתמש במשחק

שלב ראשון:

(Config.txt)יש להיכנס לתיקיית המשחק בה נמצאים קובץ ההרצה של המשחק וקובץ בשם

שלב שני:

פתח את קובץ הטקסט והכנס שם את האייפי של השרת (בשלב זה על מנהל השרת להעביר לך את האייפי באופן ידני כביכול)

שלב שלישי:

הרץ את קובץ ההרצה של המשחק והכנס את פרטי המשתמש שלך, אם אין לך עליך להירשם.

## חוקי המשחק

שלב ראשון – מיקום צוללות:

תחילה אתה מקבל שתי צוללות גדולות בגודל של ארבע משבצות, שלושה צוללות בגודל של שלוש משבצות, ארבעה צוללות בגודל של שתי משבצות ושלושה צוללות קטנות בגודל של משבצת אחת.

כמות הצוללות שנשארו לך למקם נמצאים תמיד בתחתית הצאט לוג שבצד שמאל.

מעל לוח המשחק ישנם את שמותם של שני המשתתפים, אם אתה שחקן מספר 1 אתה ממקם את הצוללות בצד שמאל של המשחק, אם אתה שחקן מספר 2 אתה ממקם את הצוללות בצד ימין, לכל אחד יש שטח של 200 משבצות (10 על 20) בו הוא יכול למקם את הצוללות.

מיקום הצוללות מתבצע רק לרוחב המגרש. על מנת למקם צוללת עליך לתפוס שתי נקודות לרוחב שביניהם אתה רוצה למקם את הצוללת, משמעות של דבר זה היא שעל מנת למקם את הצוללת אתה צריך ללחוץ פעמים, פעם אחת בהתחלה ופעם אחת בסוף (או הפוך).

הסרת הצוללת היא אופציה הניתנת לך רק אם לא אישרת שסיימת למקם את הצוללות, בכדי לעשות זאת, אתה צריך ללחוץ פעמיים על צוללת ובכך היא תוסר. כמובן שאם אתה מסיר צוללת היא חוזרת אליך בכך שתוכל למקם אותה במקום אחר.

לאחר שסיימת למקם את הצוללות עליך ללחוץ על כפתור האנטר שבתחתית הצאט לוג שנמצא בשמאל.

שלב שתיים – פגיעה ביריב:

שלב זה מתחיל לאחר ששתי הצדדים אישרו שהם סיימו למקם את צוללותיהם. בשלב זה המטרה היא לפוצץ את כל הצוללות שהיריב מיקם. כל שחקן בתורו מנסה לפגוע ביריב, אם פגעת ביריב תוכל לראות את נקודת הפגיעה מסומנת בשחור חזק, אם החטאתה תוכל לראות את הנקודה מסומנת בשחור חלש, כל פגיעה של היריב תתועד על המגרש שלך בצבע אדום (גם אם היריב החטיא)

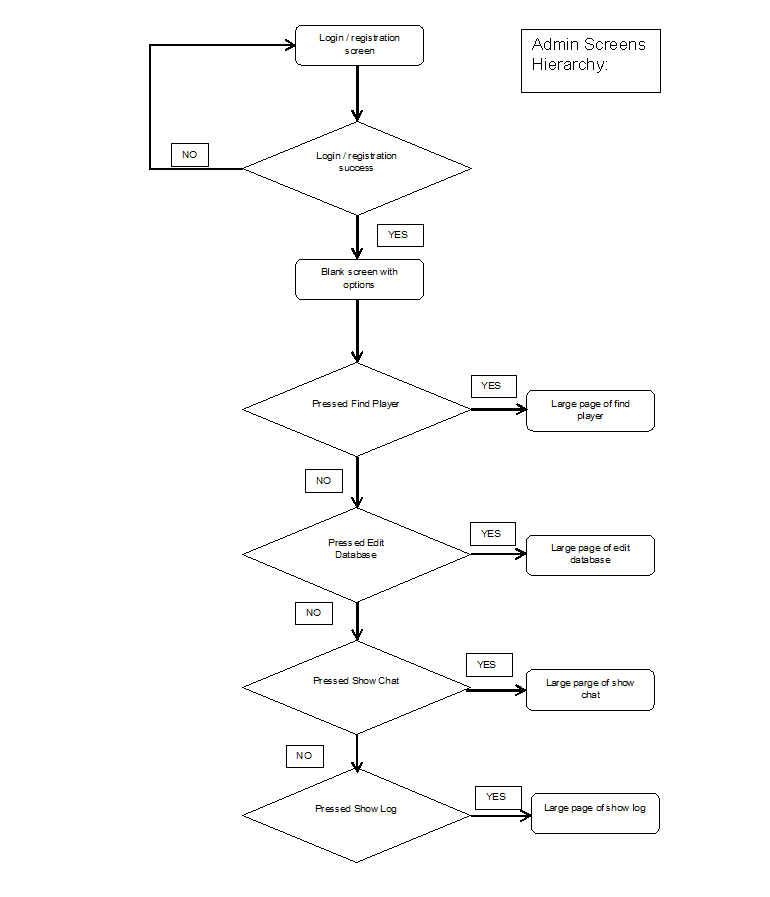
בנוסף לכך כל דבר כמו תחילת תור, פגיעה, החטאה, ניצחון והפסד תוכל לראות בצאט לוג שנמצא בצד שמאל.

שלב שלוש – סיום משחק:

במקרה שניצחת או הפסדת תקבל על כך הודעה בצאט לוג, עליך יהיה ללחוץ על כפתור האנטר על מנת לסיים את המשחק ולעבור למסך הסיכום.

[קישור לסרטון חוקי המשחק](https://youtu.be/ZiixGl4ZH6s)

## היררכית מסכים



## הסברים על חלונות

רקע קצר

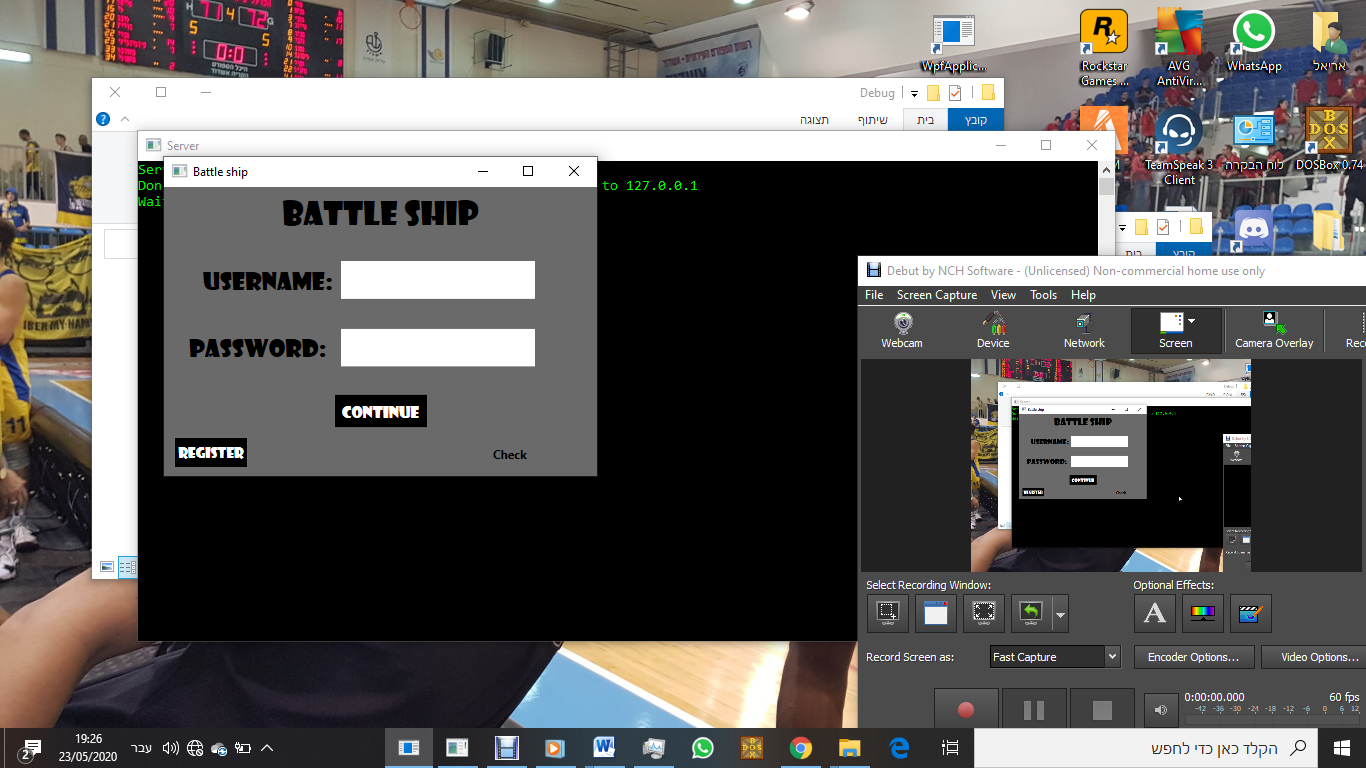
על כל אחד מהחלונות הבאים יש אופציה ללחוץ על לחצן היציאה שיגרום לסגירה מידית של המשחק.

משחק:

חלון כניסה:

חלון הכניסה הוא החלון הראשון שרואים כשפותחים את המשחק.

בחלון זה נצטרך למלא את פרטי המשתמש שלנו, תחילה נמלא את שם המשתמש ולאחר מכן את הסיסמא, אם אין לנו משתמש רשום, נוכל למלא פרטי משתמש חדש וללחוץ על לחצן ההרשמה, אם שם המשתמש כבר יהיה רשום המערכת תבקש להזין שם משתמש חדש.



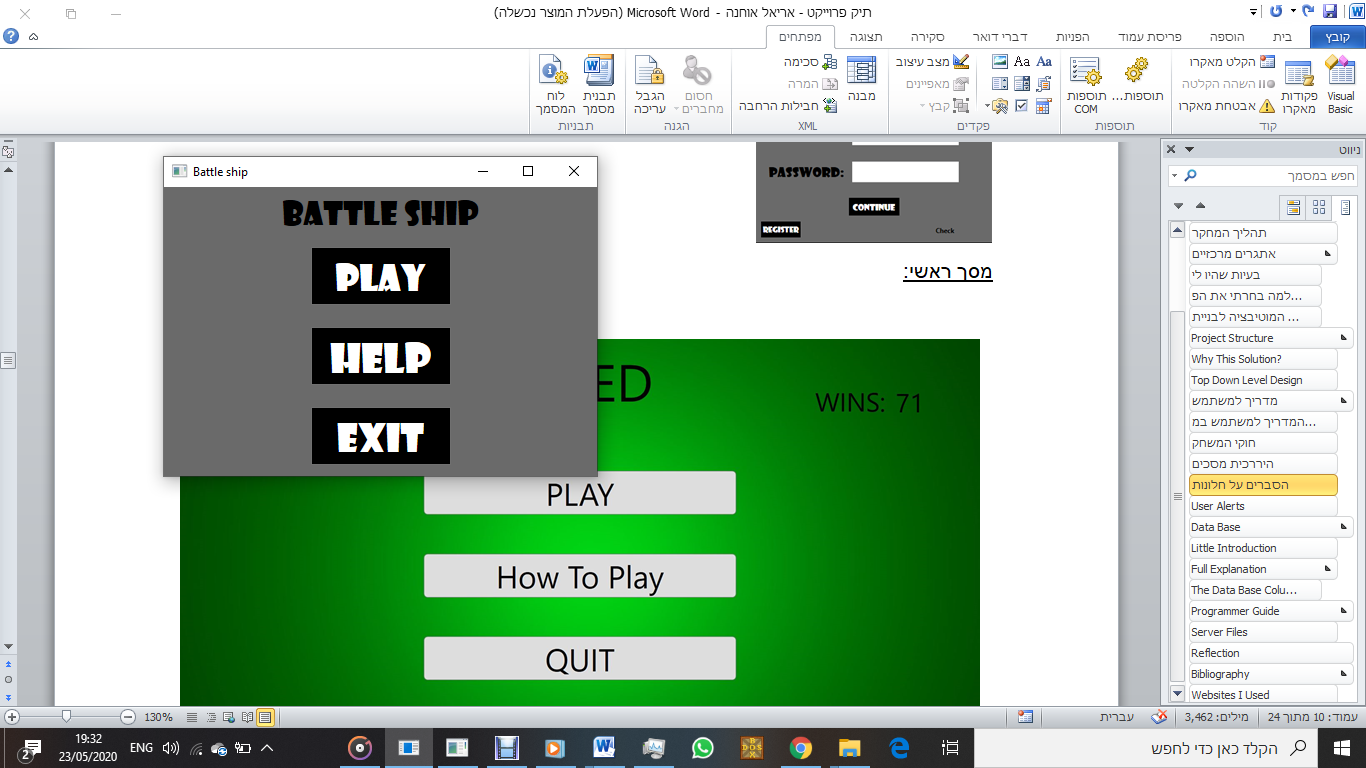
מסך ראשי:

במשחק זה נוכל לבחור בין שלושה אופציות

Play –התחלת משחק, המערכת תשהה את עצמה עד שתמצא משתמש ותפתח את חלון המשחק

Help – נועד בכדי לעזור למשתמשים, אופציה זו תפתח את תמצות חוקי המשחק ותאפשר למשתמשים לשלוח הודעות למנהלי המשחק אם נפגעו או רומו במהלך המשחק.

Exit – יציאה מהמשחק

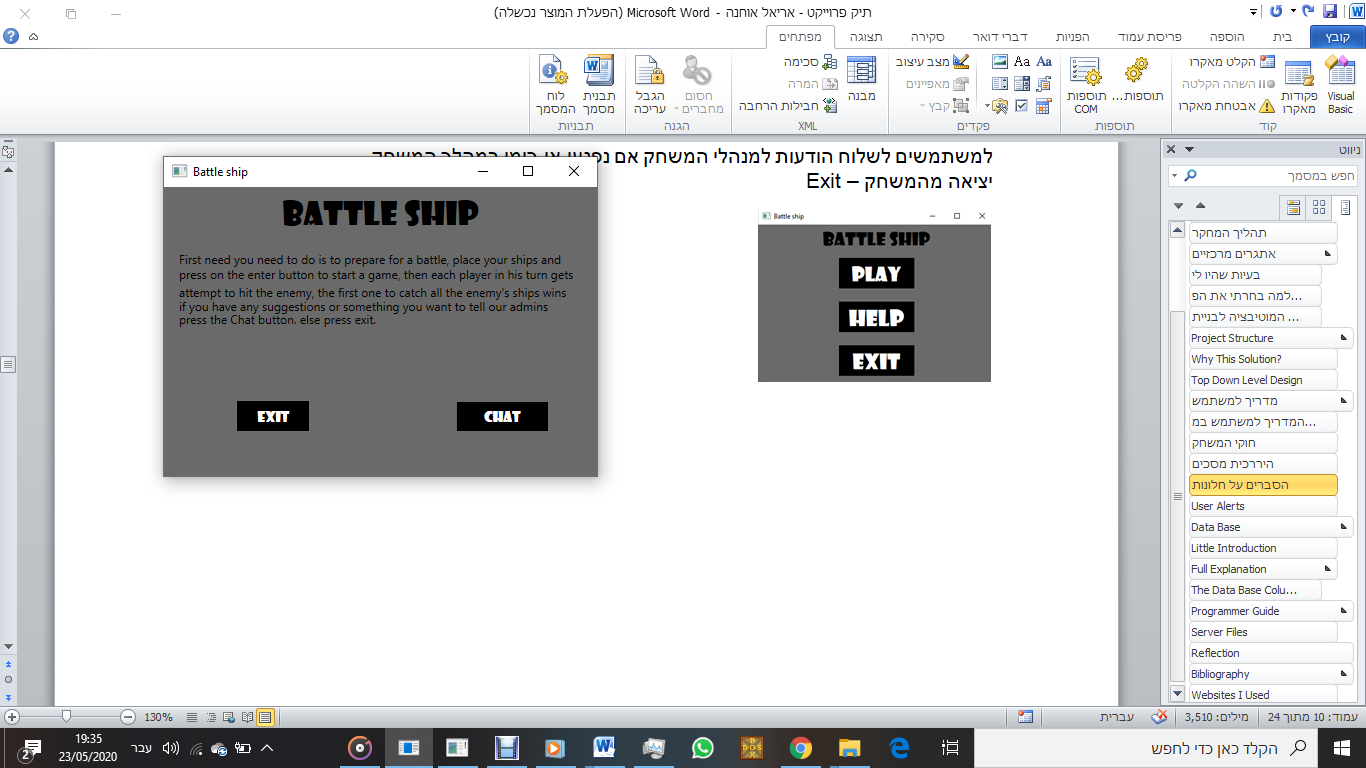


:Help מסך

מסך זה כפי שנכתב נועד בכדי לעזור למשתמשים להבין את החוקים או כדי להתכתב עם מנהלי המשחק

Chat - בכדי להתכתב

Exit – בכדי לצאת חזרה למסך הראשי

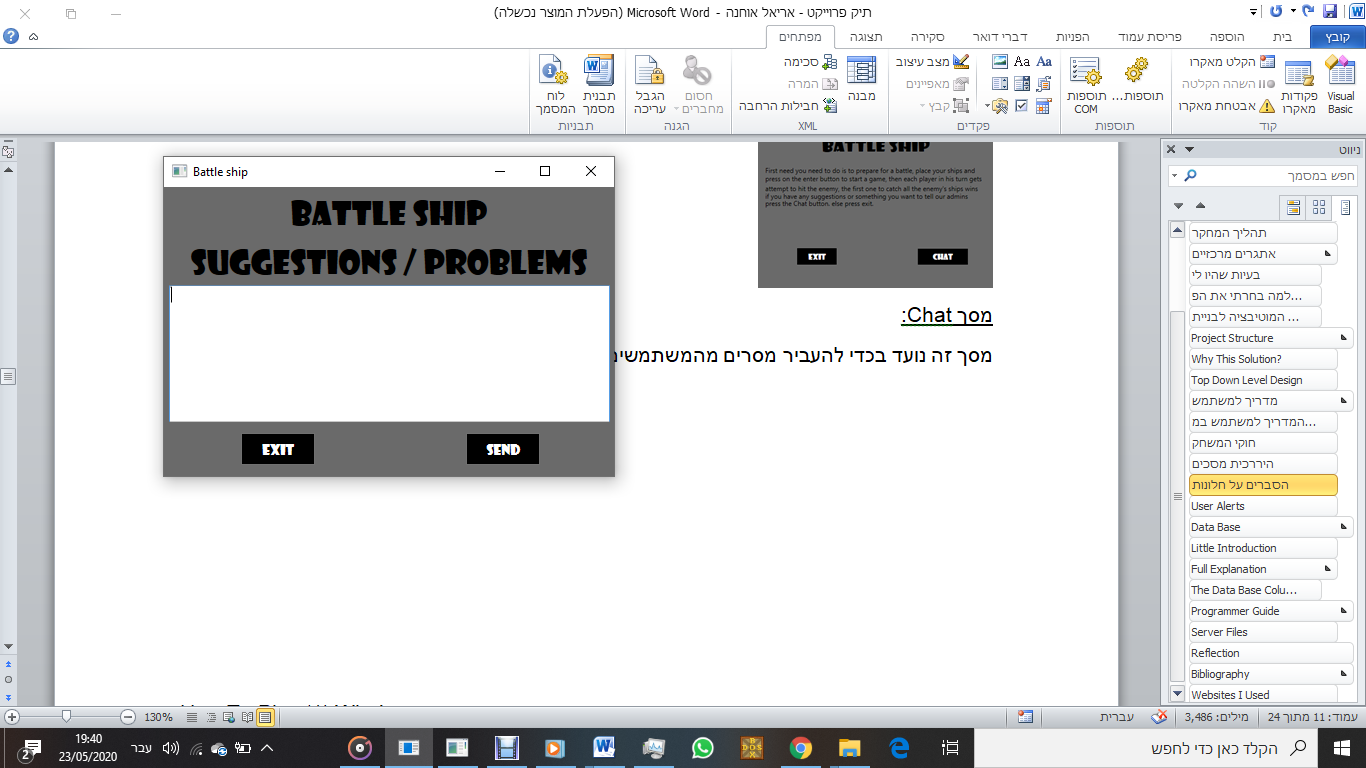


:Chat מסך

מסך זה נועד בכדי להעביר מסרים מהמשתמשים הרגילים למנהלי המשחק

Send – לשלוח את המכתב למנהלים

Exit – לצאת חזרה למסך הראשי



מסך המשחק:

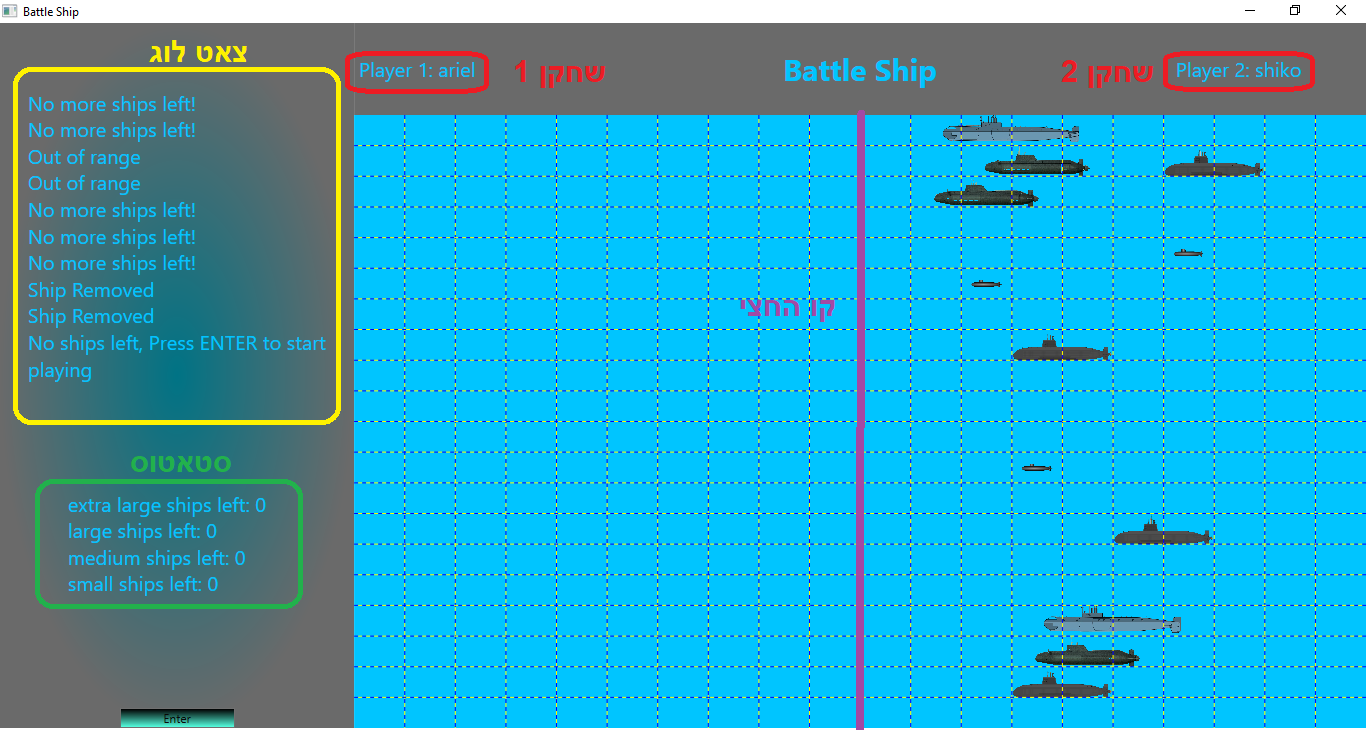
באדום מסומנים שמות השחקנים

בצהוב מסומן הצאט לוג – מטרת הצאט לוג היא לתאר את מה שקורה במשחק בכדי שהשחקן ידע יותר טוב מה קרה וקורה במשחק.

בירוק מסומן הסטאטוס – בסטאטוס רשום את הדברים החשובים ביותר, בחלק הראשון של מיקום הצוללות רשום שם את כמות הצוללות שיש למקם ומאיזה סוג, לאחר מכן אחרי מיקום הצוללות בשלב הפגיעות רשום שם כמה פגיעות נותרו לשחקן בכדי לנצח

בירוק מסומן קו החצי – המחלק את המגרש לחלקים של שני השחקנים

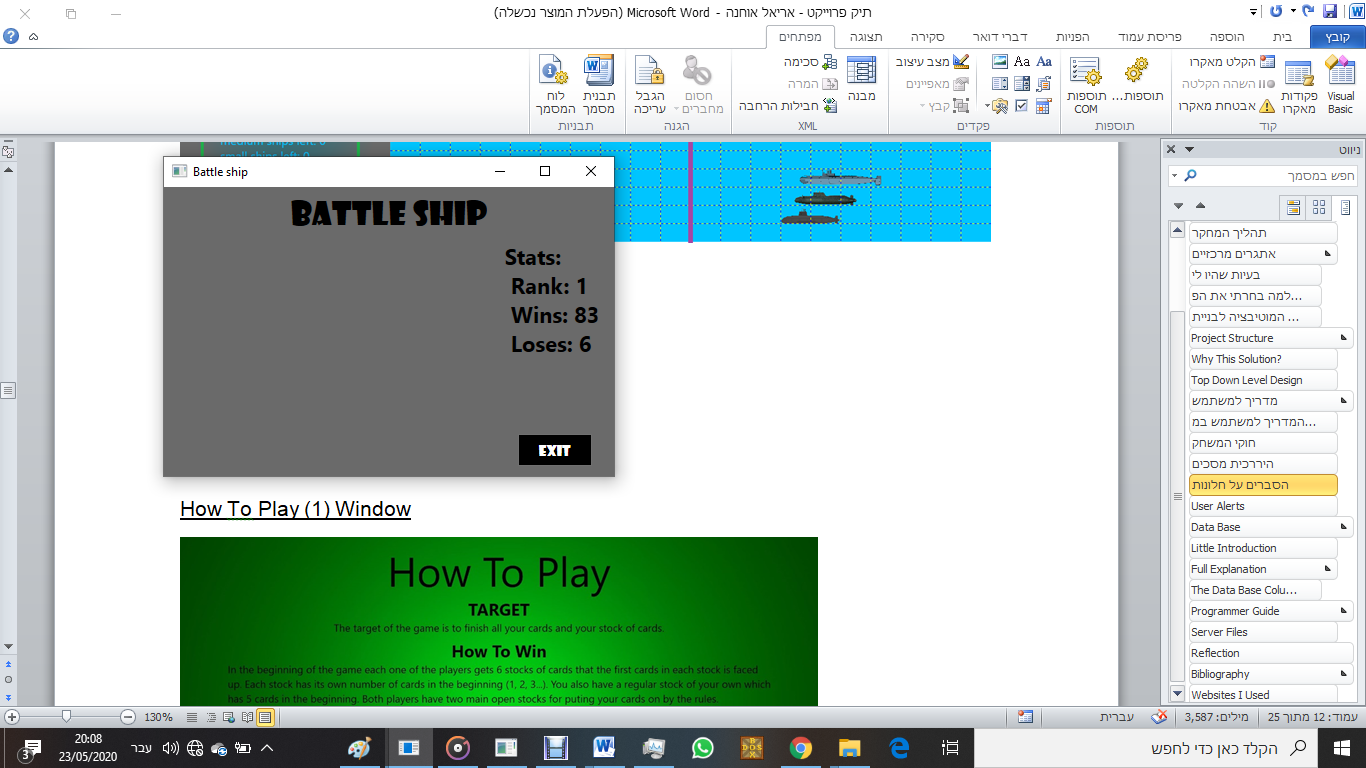
למטה בצד שמאל ישנו הכפתור אנטר – מטרתו היא לאשר סיום מיקום צוללות וסיום משחק.



מסך סיום משחק:

מטרת מסך זה היא לסכם את נתוני שחקן, רמת האדמין שלו, כמות הנצחונות וכמות ההפסדים שלו.

Exitלחץ על כפתור זה בכדי לצאת למסך הראשי -

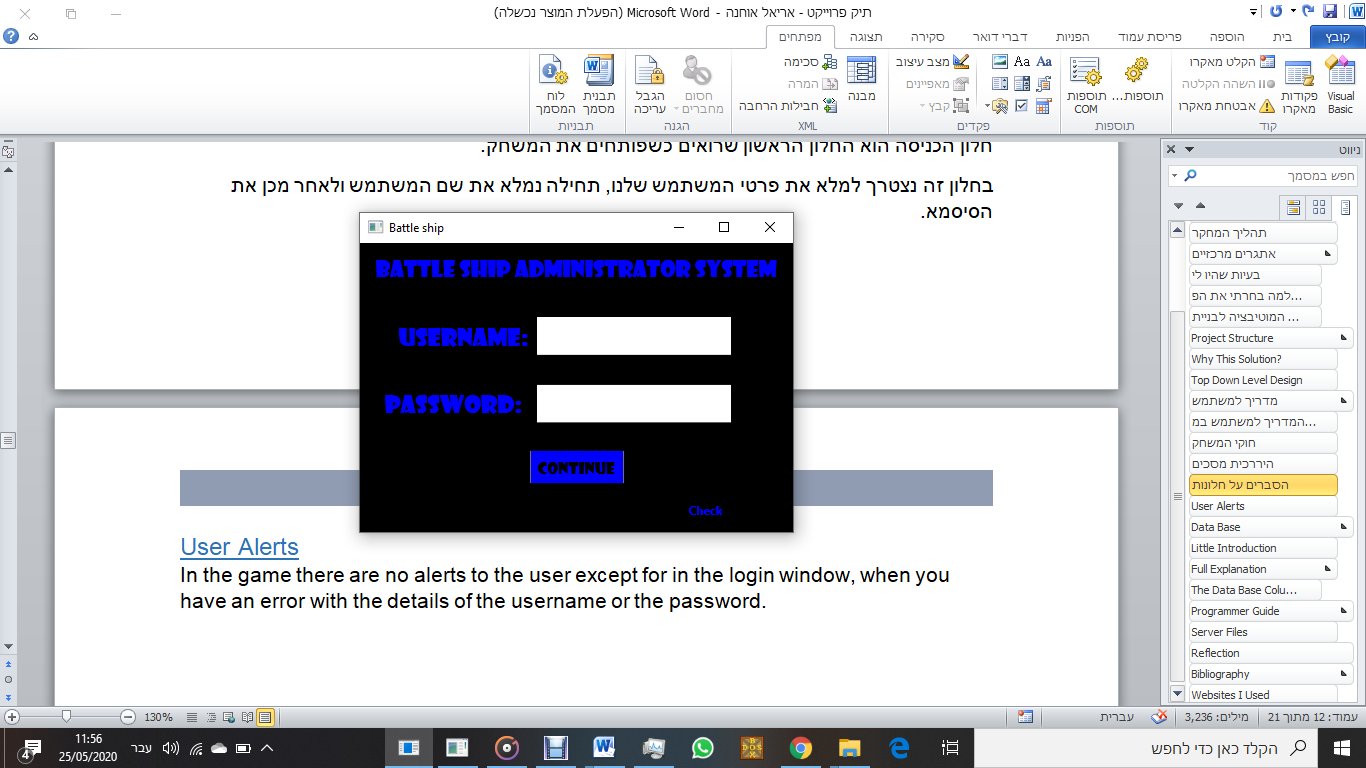


מערכת מנהלים

חלון כניסה:

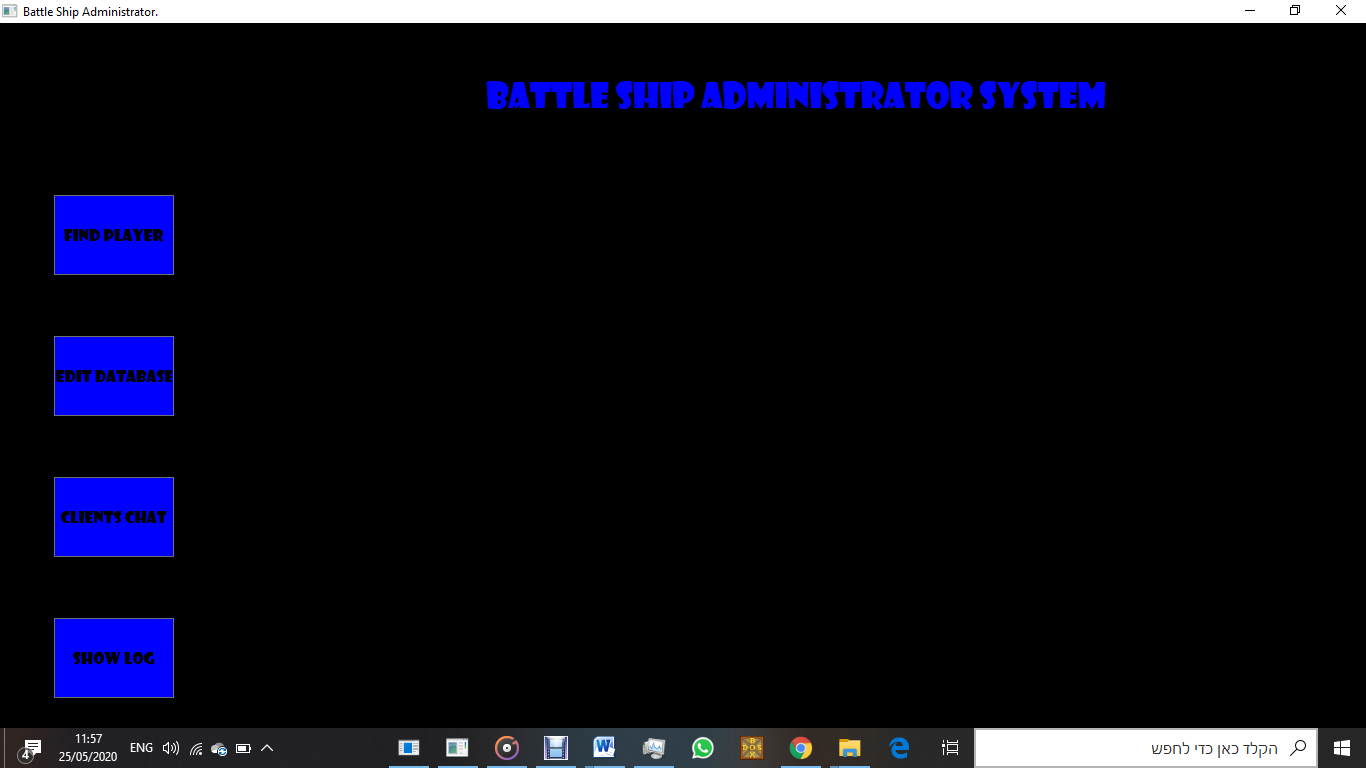
חלון הכניסה הוא החלון הראשון שרואים כשפותחים את המשחק.

בחלון זה נצטרך למלא את פרטי המשתמש שלנו, תחילה נמלא את שם המשתמש ולאחר מכן את הסיסמא.



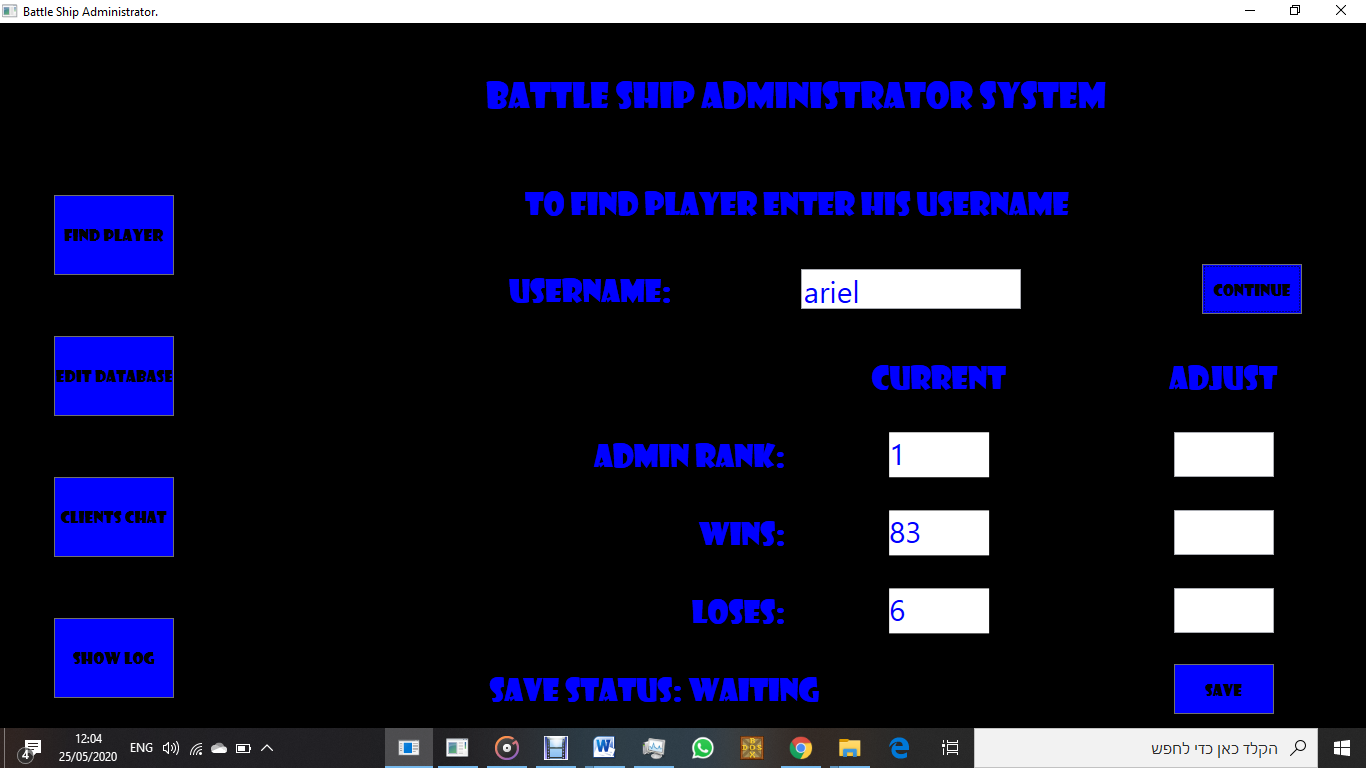
חלון ראשי:

חלון זה הוא בעצם המסך היחיד שנמצא במערכת המנהלים ומחולק לשתי גרידים, בגריד השמאלי ישנן ההגדרות, ובצד הימני הגריד המרכזי שעובדים עליו, משמע אם אני בוחר לדוגמא בעריכת מסד הנתונים מסך עריכת הנתונים יופיע לי בצד ימין.



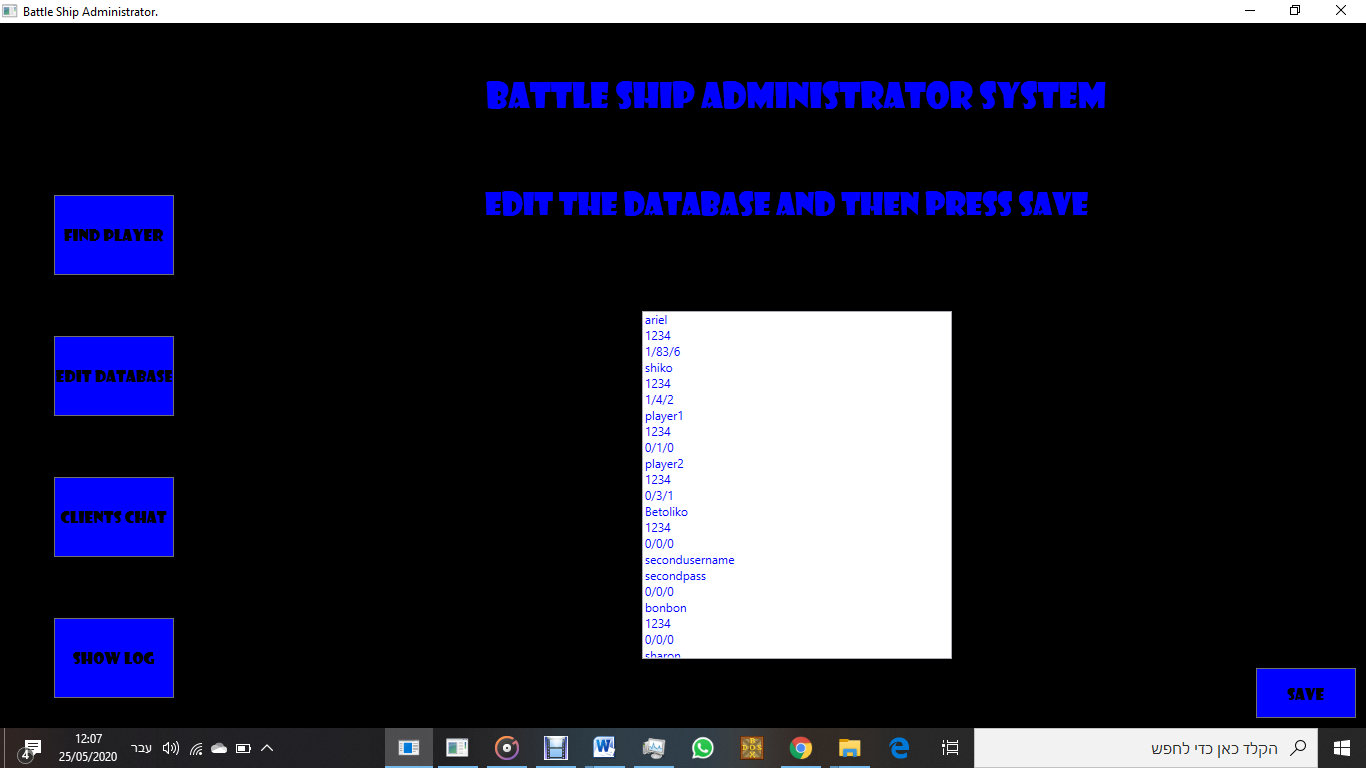
Find Player –מסך ראשון

מסך זה אחראי על חיפוש השחקנים, משמע אם יש משתמש שאני רוצה לערוך את נתוניו באופן מסודר ואחראי אני אחפש אותו שם, ואם הוא באמת קיים אני אקבל מהשרת את הנתונים שלו ואוכל לערוך אותם מבלי חשש לפגוע במבנה מסד הנתונים.



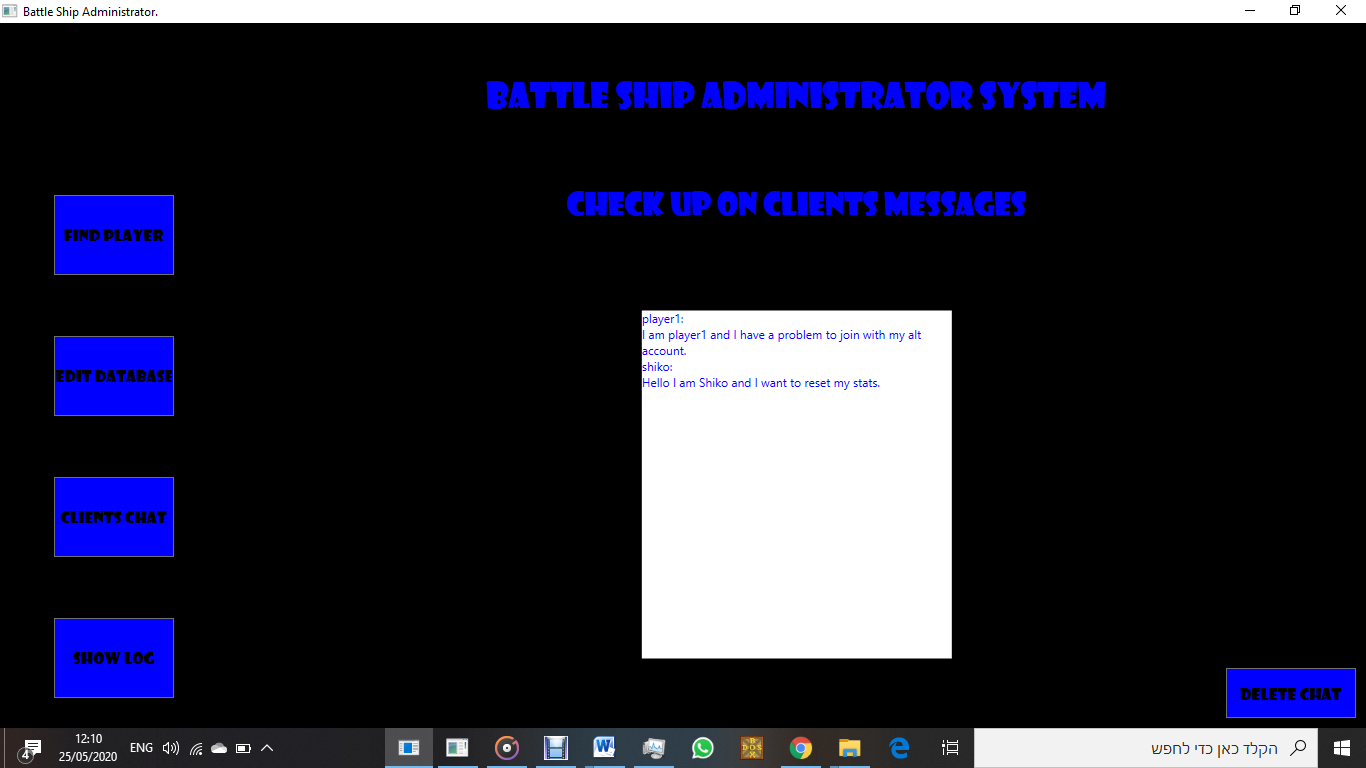
Edit database – מסך שני

מסך זה הוא בעיקרון דומה למסך הראשון עם שינוי קטן, מסך זה הוא בעל יותר אופציות שינוי מכוון שדרכו אפשר לשנות כל דבר במשתמש כמו השם משתמש והסיסמא שלו ואפילו להוסיף ולמחוק משתמשים, אך לדברים אלו יש סיכון מכוון שעריכה לא לפי הפורמט המקובל יכולה לסבך את כל מערכת המשתמשים.



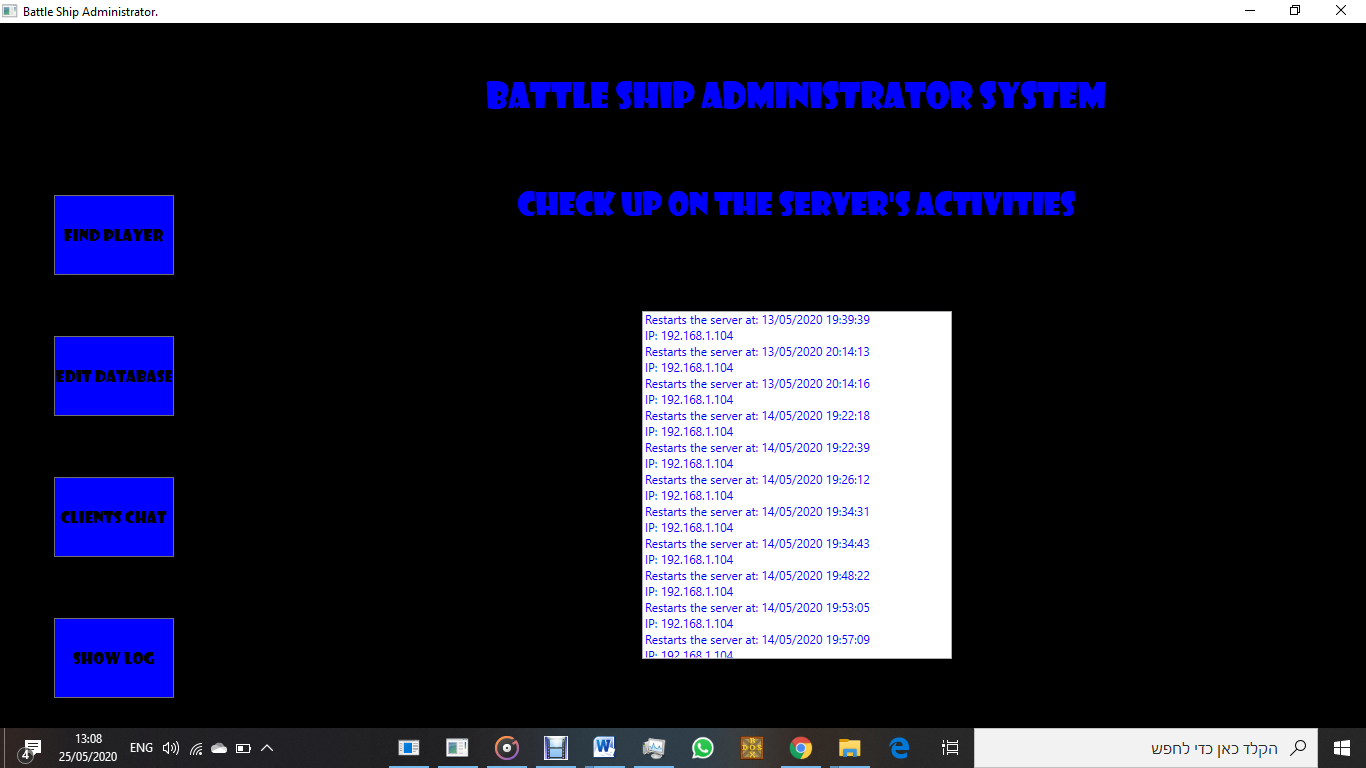
Clients Chat – מסך שלישי

מסך זה אחראי על התקשורת בין השחקנים למנהלי המשחק, במסך זה המנהלים של המשחק רואים את כל חוות הדעת והבקשות של השחקנים, דבר זה יעיל ליצירת קשר חזק עם השחקנים והופך את המשחק למשהו יותר רציני מסתם משחק.



Show Log – מסך רביעי

מסך זה אחראי על הגדרות השרת, בכדי שמנהלי השרת ידעו שיש להם שליטה מלאה על המשחק המסך מציג את כל הפעמים בהם הורץ השרת ומאיזה כתובת אייפי, דבר זה עוזר למנהלים לזהות אם אדם אחר הפעיל את השרת.



# מאגר נתונים

## הסבר על הקבצים.

מאגר הנתונים בפרויקט שלי מחולק לארבעה קבצים מסוג טקסט, באמצעות קבצים אלו אני שומר את כל הנתונים.

הקבצים הם:

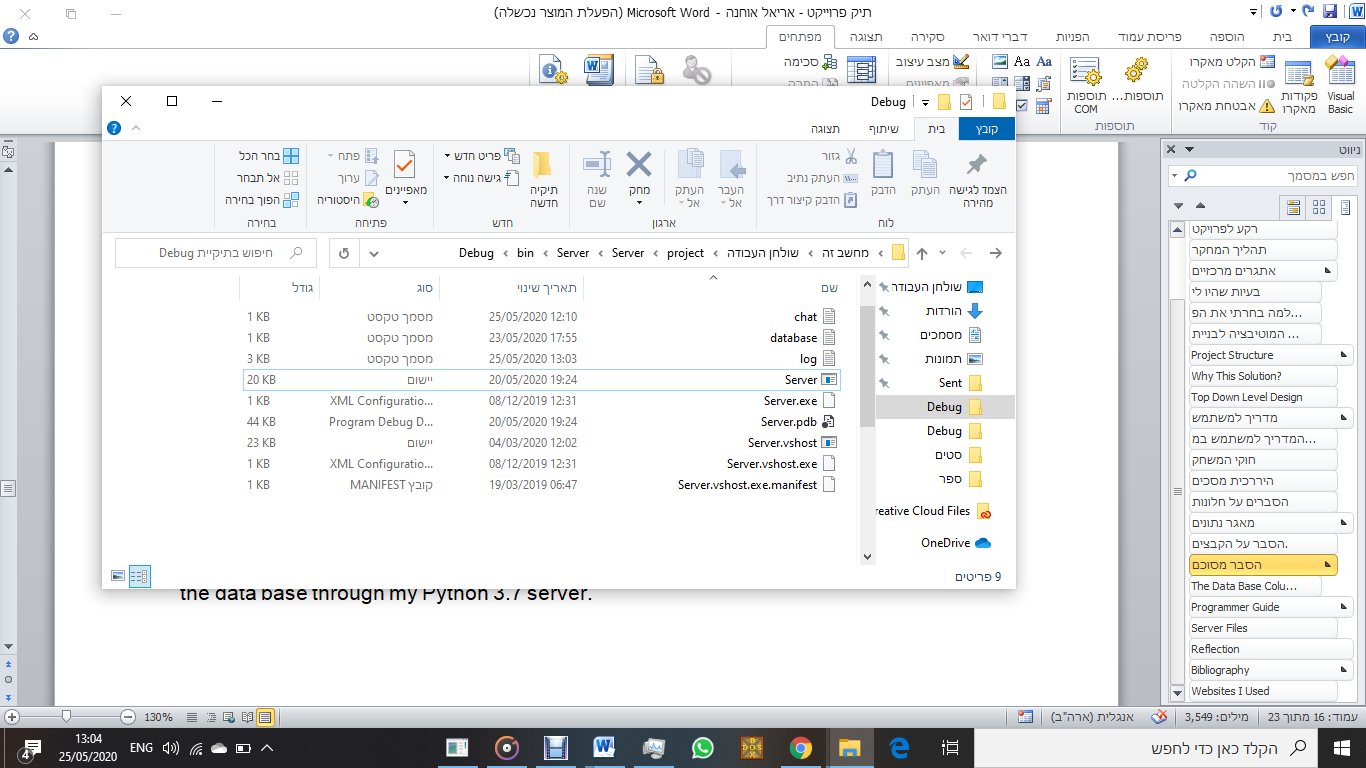
Log.txt – הקובץ מכיל את נתוני הפעלת השרת, תאריך שעה ואייפי שהריץ את השרת, קובץ זה משומש רק על ידי מנהלי המשחק באמצעות מערכת המנהלים. קובץ זה נשמר בתיקיית השרת

Chat.txt – קובץ זה אחראי על העברת המסרים מהלקוחות למנהלי המשחק, באמצעות קובץ זה לקוחות המשחק מעבירים מסרים למנהלי המשחק. קובץ זה נשמר בתיקיית השרת

Database.txt – הקובץ החשוב ביותר, קובץ זה אחראי על שמירת נתוני השחקנים לפי הפורמט בו הוא שומר לכל שחקן את שם המשתמש שלו, הסיסמא שלו, רמת המנהל שלו, כמות הניצחונות וההפסדים שלו. קובץ זה ניתן לשינוי על ידי מערכת המנהלים. קובץ זה נשמר בתיקיית השרת

Config.txt – קובץ זה נשמר בתיקיית המשחק ובתיקיית מערכת המנהלים, קובץ זה נועד בכדי שהלקוח יוכל להתחבר לשרת מבלי לשנות את המשחק עצמו, על ידי קובץ חיצוני זה המשתמש מזין בו את האייפי של השרת ובכך המשחק (או מערכת המנהלים) יודע להתחבר לשרת.

## הסבר מסוכם



כפי שנכתב, בתיקיית השרת אמורים להיות שלושה קבצים מסוג טקסט

Chat.txt

Database.txt

Log.txt

קבצים אלה נראים כך:

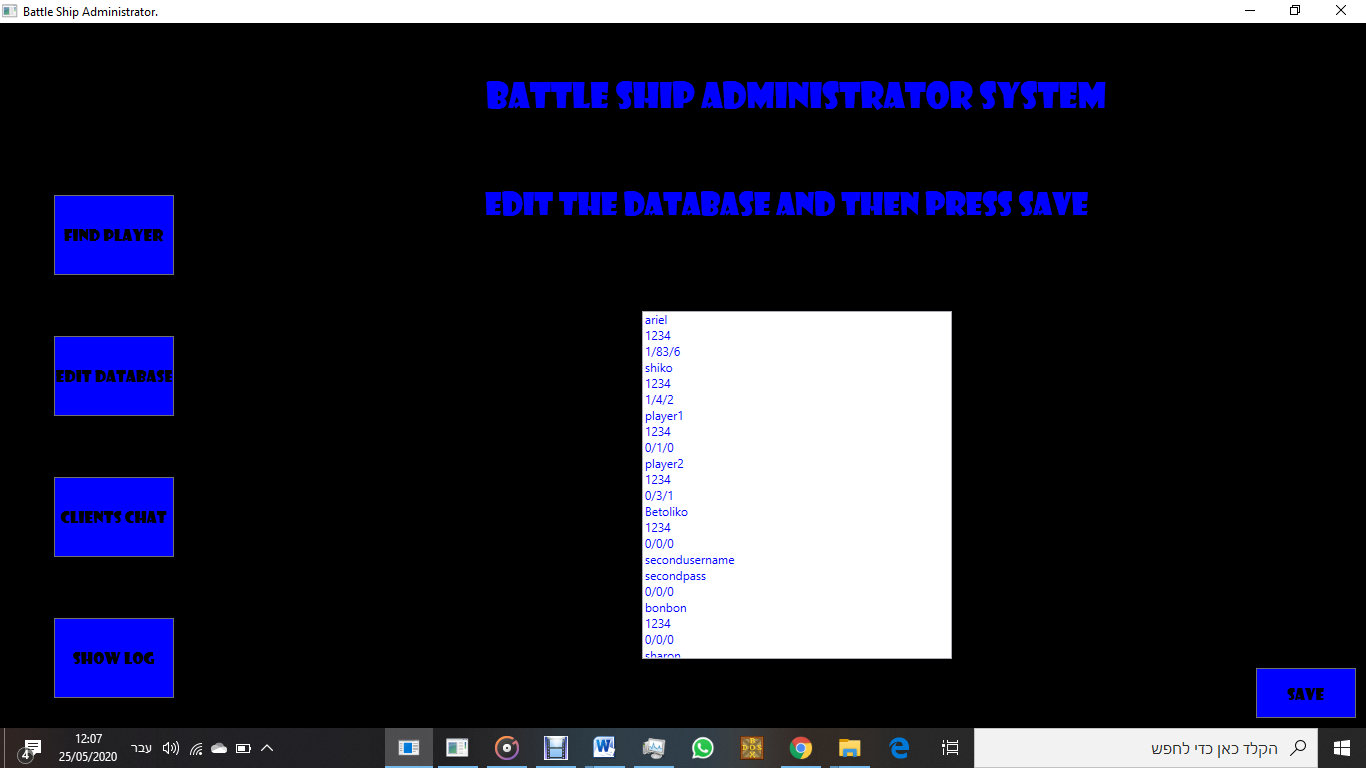
Database:

Format:

(username)

(password)

(admin rank)/(wins)/(loses)

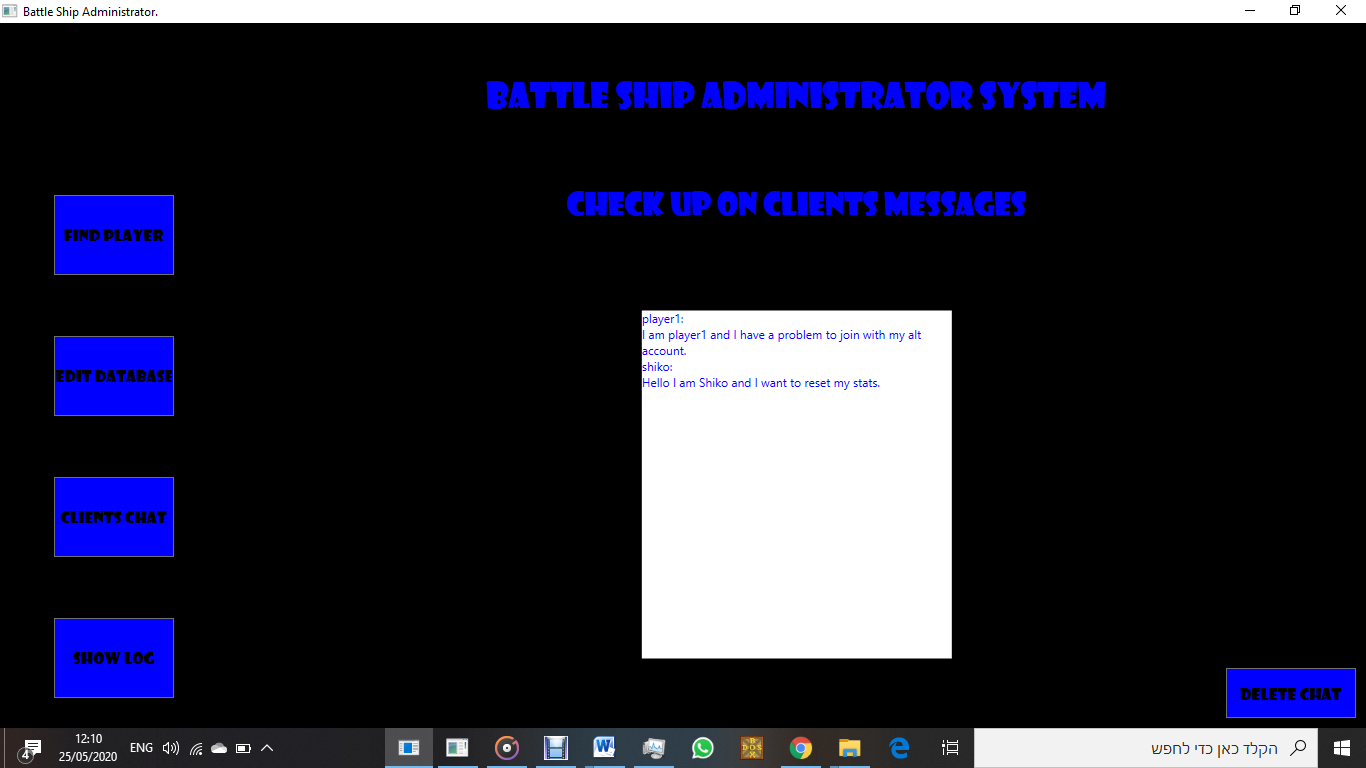


Chat.txt

Format:

(writer username):

(message)

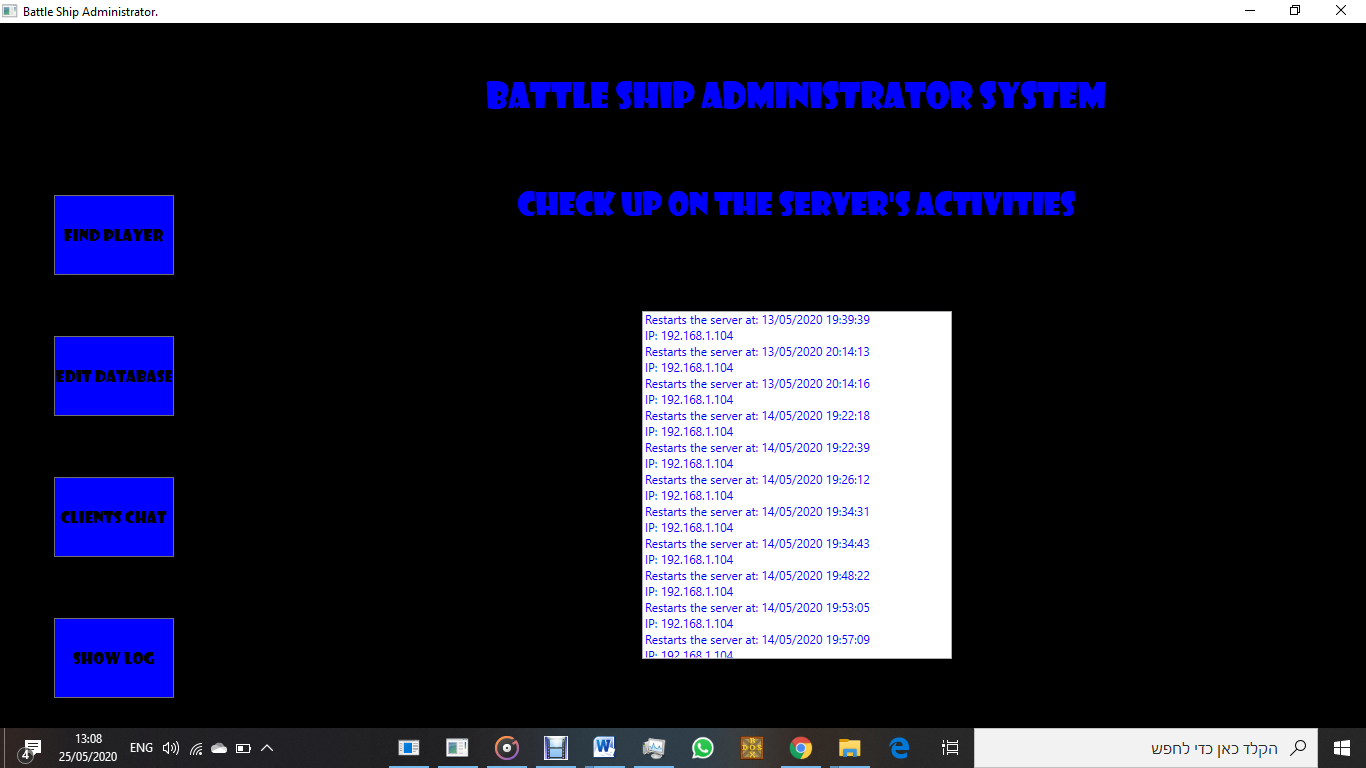


Log.txt:

Format:

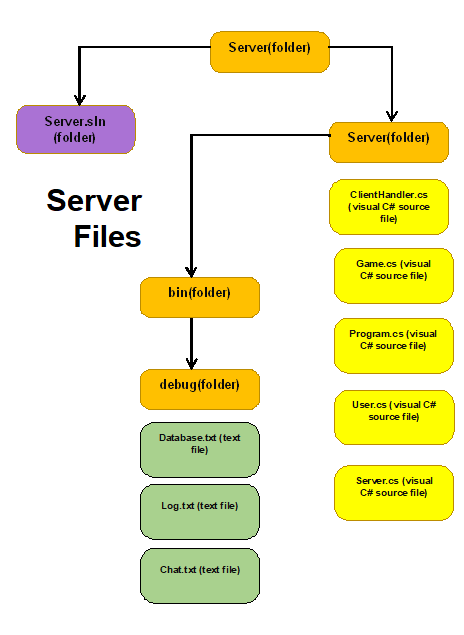
Restarts the server at: (time of restart)

IP: (IP who started the server)



# מדריך למפתח

## קבצי השרת



server.sln

הקובץ הזה הוא קובץ שנוצר על ידי וויזואל סטודיו ומשומש בשביל לכתוב את הפרויקט, הקובץ ממוקם בתוך תיקיית השרת (העליונה יותר), הקובץ עצמו משתמש במחלקות בנמצאות בתוך תיקיית השרת (הנמוכה יותר)

ClientHandler.cs

קובץ זה הוא מחלקה הממוקמת בתוך תיקייה בשם סרבר (הנמוכה יותר), מחלקה זו אחראית על טיפול בלקוח, משמע השרת יוצר לכל לקוח טיפוס מין מחלקה זו וכל טיפוס כזה אחראי על העברת נתונים בין השרת ללקוח שעליו הוא ממונה.

Code Printout – Client Handler.cs – start

using System;

using System.Collections.Generic;

using System.IO;

using System.Net;

using System.Net.Sockets;

using System.Text;

using System.Threading;

namespace Server

{

public class ClientHandler

{

/// <summary>

/// BIGGEST CLASS IN THE PROJECT!

/// Client handler handles everything the server recieves from the client and sends everything the server sends to the clients

/// Client handler handles two scipts - the player script and the admin script

/// </summary>

public static List<ClientHandler> clntList = new List<ClientHandler>(); // static list of the clienthandlers

public static List<ClientHandler> gamers = new List<ClientHandler>(); //static list of the players waiting for game

private TcpClient clnt;

private NetworkStream stream;

private byte[] data;

private int ClientSpecificNumber; // the id of the player

private bool findgame = false;

private static List<User> users = new List<User>(); // static list of users

public static int Userslogged = 0;

private static List<string> StopPreparing = new List<string>(); // static list of stop preparing players

private static List<Game> Games = new List<Game>(); // static list of games

private bool intransmission = true;

private bool confirmedplayer = false; // checks up if the player is a confirmed player to be able to run protocol.

private bool confirmedadmin = false; // checks up if the player is an admin or not, to avoid him exacute admin's commands.

public ClientHandler(TcpClient clnt) // BUILDER - creates the stream and starts a loop on the HandleRead function

{

this.Clnt = clnt;

this.stream = clnt.GetStream();

this.Data = new byte[1024];

this.ClientSpecificNumber = Userslogged;

stream.BeginRead(data, 0, data.Length, HandleRead, null);

}

/// <summary>

///Biggest class in the Project - this class stands for recieving a Code of the Protocol

///checks if the client has access to that code and starts another function / commands in target to send back the response of the server.

/// </summary>

private void HandleRead(IAsyncResult ar)

{

try

{

if (intransmission) // if the player is still logged.

{

string recived = Encoding.UTF8.GetString(data, 0, data.Length);

recived = recived.Replace("\0", "");

string msg = recived;

recived = String.Format("{0}: {1}", ((IPEndPoint)clnt.Client.RemoteEndPoint).Address.ToString(), recived);

Console.WriteLine(recived);

recived = msg;

Array.Clear(data, 0, data.Length);

if (recived.Length >= 3) // avoids exp of "string PROTOCOL = recived.Substring(0, 3);" out of range

{

string PROTOCOL = recived.Substring(0, 3);

if (PROTOCOL == "LOG")//Log in, signs in users by checking their username and password in the log file

{

if (LOGUSER(recived))//check if username and passowrd are right

{

Send("Logged in, ID: " + Userslogged);

Userslogged++;

confirmedplayer = true; // sets the player confirmed to be able to run game codes (STS, PLY, WIN for example..)

}

else

Send("Failed");

}

if (PROTOCOL == "REG")//registering new user if possible.

{

if (REGUSER(recived))

{

Send("Registered");

Userslogged++;

confirmedplayer = true; // sets the player confirmed to be able to run game codes (STS, PLY, WIN for example..)

}

else

Send("Can't Register");

}

if (PROTOCOL == "LAG") // Log Admin user - Administrator script

{

if (LOGADMINUSER(recived))//check if username and passowrd are right

{

Send("Logged in, ID: " + Userslogged);

Userslogged++;

confirmedadmin = true; // sets the player confirmed to be able to run admin codes

}

else

Send("Failed");

}

if (PROTOCOL == "OUT")

{

intransmission = false;

}

if (confirmedadmin) // ADMIN COMMANDS ONLY - ONLY ADMINS CAN REACH TO THESE CODES!

{

if (PROTOCOL == "AFP") // Admin Protocol - Find Player - Attempts to find a player and if he exists returns his details.

{

Send(AdminFindPlayer(recived)); // if the username is in the database it sends his status, else sends a message that we couldn't find him in our database.

}

if (PROTOCOL == "SET") // Sets the new stats

{

Send(AdminSetStats(recived));

}

if(PROTOCOL == "GDB") //Get DataBase

{

Send(File.ReadAllText("database.txt"));

}

if (PROTOCOL =="EDB") // Edit Database

{

EditDataBase(recived);

}

if(PROTOCOL == "GLG")// Admin Get log.

{

Send(File.ReadAllText("log.txt"));

}

if(PROTOCOL =="GCC") // Admin Get Clients Chat

{

if (File.ReadAllText("chat.txt").Length > 0) // checks if the file is empty to know what to send.

{

Send(File.ReadAllText("chat.txt"));

}

else

{

Send("Chat file is empty");

}

}

if(PROTOCOL == "DCC") // Admin Delete Client Chat.

{

File.WriteAllText("chat.txt", ""); // Sets the Chat.txt empty

}

}

if (confirmedplayer) // ALL THE FUNCTIONS FROM HERE ARE FOR PLAYERS ONLY!!!

{

if(PROTOCOL == "CHT")

{

WriteToChat(recived);

}

if (PROTOCOL == "PLY")//attempts to find another client to play with

{

bool add = true;

for (int i = 0; i < gamers.Count; i++)

{

if (clntList[ClientSpecificNumber] == gamers[i])

add = false;

}

if (add)

gamers.Add(clntList[ClientSpecificNumber]);

{

if (gamers.Count == 2)

{

string towrite = "ID: " + ClientSpecificNumber + "and ID: " + gamers[0].ClientSpecificNumber + " starts a game!";

Console.WriteLine(towrite);

Send("GSN2" + gamers[0].ClientSpecificNumber);

gamers[0].Send("GSN1" + ClientSpecificNumber);

gamers.Clear();

}

}

}

if (PROTOCOL == "STS")//Protocol Stats! Sends stats to the server.

{

Send(SetUpStats());

string towrite = "Sending stats to ID: " + ClientSpecificNumber + ", " + SetUpStats();

Console.WriteLine(towrite);

}

if (PROTOCOL == "LGF")// LGF = Leave Game Finder, Play screen. REQUIRES FIX

{

this.findgame = false;

string towrite = "ID: " + ClientSpecificNumber + " Leaving Play Screen";

Console.WriteLine(towrite);

}

if (PROTOCOL == "GEN") //GAME PROTOCOL - Get Enemy Name!

{

int enemyidGEN = int.Parse(recived.Substring(3, 1));

Send(users[enemyidGEN].GetUN());

}

if (PROTOCOL == "SPP")// STOP PREPARING PROTOCOL = Checks if both of the praticipators ready and adds a game to the Games list if both stopped preparing.

{

if (SPPFunction(recived))

{

string towrite = "ID: " + ClientSpecificNumber + " And ID: " + recived.Substring(3, 1) + " Stopped preparing and now starting to play";

Console.WriteLine(towrite);

Games.Add(new Game(ClientSpecificNumber, int.Parse(recived.Substring(3, 1))));

Send("Stop waiting: -1,-1"); // Gives the first turn to the sender

}

else

{

string towrite = "ID: " + ClientSpecificNumber + "Stopped preparing and waiting for ID: " + recived.Substring(3, 1) + " to finish preparing";

Console.WriteLine(towrite);

}

}

if (PROTOCOL == "FLD") // Get Field Protocol, sets up the game.

{

for (int i = 0; i < Games.Count; i++)

{

if (Games[i].Player1 == ClientSpecificNumber || Games[i].Player2 == ClientSpecificNumber)

{

Games[i].SetUpMap(recived.Substring(4));

if (Games[i].CountMapSets1 > 1)

{

string toshow = "";

int[,] field = Games[i].GetMap();

for (int m = 0; m < field.GetLength(1); m++)

{

for (int j = 0; j < field.GetLength(0); j++)

{

toshow += field[m, j];

}

toshow += "\n";

}

Console.WriteLine(toshow);

}

}

}

} // FLD end

if (PROTOCOL == "HIT")// Game Protocol, gets a point and check if it hits the enemy's battle ship, uses field of both players in game.

{

for (int i = 0; i < Games.Count; i++)

{

int col;

int row;

if (Games[i].Player1 == ClientSpecificNumber)

{

msg = msg.Remove(0, 5);//The protocol.

int firstnum = msg.IndexOf(","); // finds the point to split.

col = int.Parse(msg.Substring(0, firstnum)); // The column.

row = int.Parse(msg.Substring(firstnum + 1));// The row.

if (Games[i].HitMap(col, row)) // Starts the HITMAP function by sending them the col and the row I recieved.

Send("C"); // incase Caught one of the enemy's ships.

else

Send("M"); //incase missed.

clntList[Games[i].Player2].Send("Stop waiting: " + col + "," + row);//Sends the other player that's his turn to play

}

if (Games[i].Player2 == ClientSpecificNumber)

{

msg = msg.Remove(0, 5);//The protocol.

int firstnum = msg.IndexOf(","); // finds the point to split.

col = int.Parse(msg.Substring(0, firstnum)); // The column.

row = int.Parse(msg.Substring(firstnum + 1));// The row.

if (Games[i].HitMap(col, row)) // Starts the HITMAP function by sending them the col and the row I recieved.

Send("C"); // incase Caught one of the enemy's ships.

else

Send("M"); //incase missed.

clntList[Games[i].Player1].Send("Stop waiting: " + col + "," + row);//Sends the other player that's his turn to play

}

}

}

if (PROTOCOL == "WIN")// Game Protocol, updates win!

{

users[ClientSpecificNumber].AddWin(); // updates the wins

string towrite = "ID: " + ClientSpecificNumber + " won and now he has " + users[ClientSpecificNumber].GetW() + " wins";

Console.WriteLine(towrite);

//using (StreamWriter sw = new StreamWriter("log.txt", true)) // Saves "towrite" on the log file.

//{

// sw.WriteLine(towrite);

//}

for (int i = 0; i < Games.Count; i++)

{

if (Games[i].Player1 == ClientSpecificNumber)

{

Games[i].GameFinished1 = 1;//Sets the winner player 1.

users[Games[i].Player2].AddLose(); //Adds a lose for the enemy who lose the game

}

if (Games[i].Player2 == ClientSpecificNumber)

{

Games[i].GameFinished1 = 2;//Sets the winner player 2.

users[Games[i].Player1].AddLose(); //Adds a lose for the enemy who lose the game

}

}

}

if (PROTOCOL == "CHW")// Game Protocol, Check winner

{

int gameid = -1;

bool toret = false;

for (int i = 0; i < Games.Count; i++)

{

if (Games[i].Player1 == ClientSpecificNumber)

{

Thread.Sleep(500);// To avoid running before it should

if (Games[i].GameFinished1 == 2)//Checks if his enemy won

{

toret = true;

gameid = i; // sets the game id (the fact it gets here confirms that)

}

}

if (Games[i].Player2 == ClientSpecificNumber)

{

Thread.Sleep(500);// To avoid running before it should

if (Games[i].GameFinished1 == 1)//Checks if his enemy won

{

toret = true;

gameid = i; // sets the game id (the fact it gets here confirms that)

}

}

}

if (toret)

{

if(gameid != -1)//making sure game id updated!

{

Games.Remove(Games[gameid]);

}

Send("Enemy won");

string towrite = "ID: " + ClientSpecificNumber + " lose and now he has " + users[ClientSpecificNumber].GetL() + " loses";

Console.WriteLine(towrite);

}

else

{

Send("Game on");

}

}

}

}

if (intransmission)

{

stream.BeginRead(data, 0, data.Length, HandleRead, null);

}

else

{

ClientDisconnected();

}

}

}

catch (Exception Exp) // incase of exception throws the client and writes the exception.

{

Console.WriteLine(Exp);

ClientDisconnected();

}

}

/// <summary>

/// Sets up the stats to send

/// </summary>

/// <returns>Returns the stats of the player</returns>

public string SetUpStats()

{

string tosend = "";

tosend += users[ClientSpecificNumber].GetR() + "/"+users[ClientSpecificNumber].GetW() + "/"+ users[ClientSpecificNumber].GetL();

return tosend;

}

/// <summary>

/// Send function, the function that sends the data to the client.

/// </summary>

/// <param name="msg">Sends "msg" to the client.</param>

private void Send(string msg)

{

try

{

byte[] toSend = Encoding.UTF8.GetBytes(msg);

stream.Write(toSend, 0, toSend.Length);

stream.Flush(); // cleans the stream.

string towrite = String.Format("{0}: {1}", ((IPEndPoint)clnt.Client.RemoteEndPoint).Address.ToString(), msg);

Console.WriteLine(towrite);

}

catch(Exception)

{

ClientDisconnected();

}

}

/// <summary>

/// log user, the function that signing in the users.

/// </summary>

/// <param name="msg">"msg" is the string of the username and the password.</param>

/// <returns> Returns a boolean value if the client logged or not.</returns>

private bool LOGUSER(string msg)

{

bool tosend = false;

string[] CodeLines = File.ReadAllLines("database.txt");

msg =msg.Remove(0, 6);

string givenusername = msg.Substring(0, msg.IndexOf("\n"));

string givenpassword = msg.Substring(givenusername.Length + 1, msg.Length- givenusername.Length-2);

bool searching = true;

for (int i = 0; i < CodeLines.Length - 1 && searching; i = i + 3)

{

if (givenusername == CodeLines[i])

{

if (CodeLines[i + 1] == givenpassword)

{

tosend = true;

searching = false;

users.Add(new User(givenusername, i, Userslogged));

string towrite = givenusername + "'s stats: \n ID: "+Userslogged + "\n Admin Rank: " + users[Userslogged].GetR()+ "\n Wins: "+users[Userslogged].GetW() +"\n Loses: " + users[Userslogged].GetL();

Console.WriteLine(towrite);

}

}

}

return tosend;

}

/// <summary>

/// Register user, the function that signing up the new users.

/// </summary>

/// <param name="msg">"msg" is a string of the username and password connected</param>

/// <returns> returns a boolean value if the client can register with the given username.</returns>

public bool REGUSER(string msg)

{

string[] CodeLines = File.ReadAllLines("database.txt");

msg = msg.Remove(0, 6);

string givenusername = msg.Substring(0, msg.IndexOf("\n"));

string givenpassword = msg.Substring(givenusername.Length + 1, msg.Length - givenusername.Length - 2);

bool searching = true;

for (int i = 0; i < CodeLines.Length - 1 && searching; i = i + 3)

{

if (givenusername == CodeLines[i])

searching = false; // user exists

}

if (searching)

{

using (StreamWriter sw = new StreamWriter("database.txt", true)) // making new user

{

sw.WriteLine(givenusername);

sw.WriteLine(givenpassword);

sw.WriteLine("0/0/0");

}

users.Add(new User(givenusername, CodeLines.Length, Userslogged));

string towrite = givenusername + "'s stats: \n ID: " + Userslogged + "\n Admin Rank: " + users[Userslogged].GetR() + "\n Wins: " + users[Userslogged].GetW() + "\n Loses: " + users[Userslogged].GetL();

Console.WriteLine(towrite);

}

return searching;

}

/// <summary>

/// Admin log in system, checks if the user is an admin and returns a boolean value if yes or no.

/// </summary>

/// <param name="msg">"msg" is the string of the username and the password.</param>

/// <returns> Returns a boolean value if the admin logged or not.</returns>

private bool LOGADMINUSER(string msg)

{

bool tosend = false;

string[] CodeLines = File.ReadAllLines("database.txt");

msg = msg.Remove(0, 6);

string givenusername = msg.Substring(0, msg.IndexOf("\n"));

string givenpassword = msg.Substring(givenusername.Length + 1, msg.Length - givenusername.Length - 2);

bool searching = true;

for (int i = 0; i < CodeLines.Length - 1 && searching; i = i + 3)

{

if (givenusername == CodeLines[i]) // checks if the username exists.

{

if (CodeLines[i + 1] == givenpassword)// if the username exists, checks his password.

{

users.Add(new User(givenusername, i, Userslogged)); // Adds a user to check if he is an admin

if (users[Userslogged].GetR() != 0) // confirms that the player isn't a normal user. checks if he is an admin

{

tosend = true;

searching = false;

string towrite = givenusername + "'s stats: \n ID: " + Userslogged + "\n Admin Rank: " + users[Userslogged].GetR() + "\n Wins: " + users[Userslogged].GetW() + "\n Loses: " + users[Userslogged].GetL();

Console.WriteLine(towrite);

}

else

{

users.Remove(users[Userslogged]);// If the player isn't an admin, removes his to avoid doubles..

}

}

}

}

return tosend;

}

/// <summary>

/// ADMIN COMMAND: runs a search on a player in the database :

/// if found sends his details but and if not send that the server couldn't find the username on the database

/// </summary>

/// <param name="msg"> "msg" is the username the admin search for.</param>

/// <returns>if the username is exists in database it sends his details(rank/wins/loses) else it will send that he is not in the system</returns>

private string AdminFindPlayer(string msg)

{

string towrite = "";

string[] CodeLines = File.ReadAllLines("database.txt");

msg = msg.Remove(0, 6);

string givenusername = msg;

bool searching = true;

for (int i = 0; i < CodeLines.Length - 1 && searching; i = i + 3)

{

if (givenusername == CodeLines[i]) // checks if the username exists.

{

users.Add(new User(givenusername, i, Userslogged)); // Adds a user to check if he is an admin

towrite = "Player Found: "+ givenusername + "'s stats: \n ID: " + Userslogged + "\n Admin Rank: " + users[Userslogged].GetR() + "\n Wins: " + users[Userslogged].GetW() + "\n Loses: " + users[Userslogged].GetL();

Console.WriteLine(towrite);

string toret = users[Userslogged].GetR() + "/" + users[Userslogged].GetW() + "/" + users[Userslogged].GetL();

users.Remove(users[Userslogged]);

return toret;

}

}

towrite = "Couldn't find " + givenusername + " in our database.";

Console.WriteLine(towrite);

return towrite;

}

/// <summary>

/// ADMIN COMMAND: the admin sets new rank/wins/loses to the user and sets Done.

/// if the username isn't found sends Couldn't find the user, if there was a problem except of finding the username

/// ((like wrong type of var (string and not int for example))) the server sends there was a problem.

/// </summary>

/// <param name="msg"> "msg" is the given details in the following form (username - rank - wins - loses) username stands

/// find the user and the rank wins and loses are to edit in his profile.</param>

/// <returns> returns if the process successed or not.</returns>

private string AdminSetStats(string msg)

{

try

{

string[] CodeLines = File.ReadAllLines("database.txt");

msg = msg.Remove(0, 6);

string givenusername = msg.Substring(0, msg.IndexOf("\n")); // cuts the username.

msg = msg.Remove(0, givenusername.Length);

int rank = int.Parse(msg.Substring(0, msg.IndexOf("/"))); // cuts the rank

int wins = int.Parse(msg.Substring(msg.IndexOf("/") + 1, msg.LastIndexOf("/") - (msg.IndexOf("/") + 1))); // cuts the wins

int loses = int.Parse(msg.Substring(msg.LastIndexOf("/") + 1)); //cuts the loses

bool searching = true;

for (int i = 0; i < CodeLines.Length - 1 && searching; i = i + 3)

{

if (givenusername == CodeLines[i]) // checks if the username exists.

{

users.Add(new User(givenusername, i, Userslogged)); // Adds a user to check if he is an admin

users[Userslogged].SetRank(rank);

users[Userslogged].SetWin(wins);

users[Userslogged].SetLose(loses);

users.Remove(users[Userslogged]); // to avoid duplicates.

return "Done";

}

}

return "Couldn't find the user";

}

catch

{

return "There was a problem";

}

}

/// <summary>

/// player command - the server recieves a message from player and sets it at chat.txt file.

/// </summary>

/// <param name="msg">the message to write.</param>

public void WriteToChat(string msg)

{

msg = msg.Remove(0, 5);

using (StreamWriter sw = new StreamWriter("chat.txt", true)) // Saves "towrite" on the chat file.

{

sw.WriteLine(msg);

}

string towrite = "Wrote to chat.txt: " + msg;

Console.WriteLine(towrite);

}

/// <summary>

/// edits the database by recieving new one.

/// </summary>

/// <param name="msg"> the edited database.</param>

public void EditDataBase(string msg)

{

msg = msg.Remove(0, 6);

File.WriteAllText("database.txt", msg);

}

/// <summary>

/// Stop Preparing function, this function checks if both of the players are ready to play,

/// if they are, it sends okay and starting a game on Games list

/// </summary>

/// <param name="recived"> The form of(the enemy's ID) (-) (this client ID) </param>

/// <returns> returns if the game started</returns>

public bool SPPFunction(string recived)

{

string firstplayer = recived.Substring(3, 1);

string msg = firstplayer;

msg = msg + "-" + ClientSpecificNumber;

if (StopPreparing.Contains(ClientSpecificNumber + "-" + firstplayer))

{

clntList[int.Parse(firstplayer)].Send("Start Playing");

Send("Start Playing");

StopPreparing.Remove(ClientSpecificNumber + "-" + firstplayer);

return true;

}

else if (StopPreparing.Contains(msg))

{

string towrite = "ID: " + ClientSpecificNumber + " Already started the SPP protocol.";

Console.WriteLine(towrite);

}

else

StopPreparing.Add(msg);

return false;

}

/// <summary>

/// Client Disconnected, disconnecting from the client and it's stream

/// </summary>

public void ClientDisconnected()

{

string towrite = "ID: " + ClientSpecificNumber + " has disconnected!";

Console.WriteLine(towrite);

clnt.Close();

stream.Close();

}

public TcpClient Clnt

{

get

{

return clnt;

}

set

{

clnt = value;

}

}

public byte[] Data

{

get

{

return data;

}

set

{

data = value;

}

}

public bool Findgame

{

get

{

return findgame;

}

set

{

findgame = value;

}

}

public int ClientSpecificNumber1

{

get

{

return ClientSpecificNumber;

}

set

{

ClientSpecificNumber = value;

}

}

}

}

Code Printout – Client Handler.cs – end.

Game.cs

קובץ זה הוא מחלקה הממוקמת בתוך תיקייה בשם סרבר (הנמוכה יותר), מחלקה זו אחראית על משחק עצמו, החזקת האיידי של השחקנים ובניית המגרש, אחרי שהמחלקה בונה את המגרש והמשחק עובר לפגיעות, המשחק עובד באמצעות פונקציה מרכזית במחלקה שאחראית על בדיקת המיקום של הפגיעות ושליחה חזרה למחלקה המרכזית של השרת.

Code Printout – Game.cs – start

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Server

{ /// <summary>

/// This class responsible of setting up the game, this one recieves the players, starts the game and sets up the board.

/// after setting up the board the main responsibility turns to check if there are hits on and board and updates them

/// </summary>

class Game

{

private int player1;

private int player2;

private int[,] Map;

private int BoardWide;

private int BoardHigh;

private int CountMapSets;

private int GameFinished; // 0 no, 1 player 1 won, 2 player 2 won.

/// <summary>

/// game builder - sets up the game and the board

/// </summary>

/// <param name="player1"> the first player ID (ClientSpecificNumber)</param>

/// <param name="player2">the second player ID (ClientSpecificNumber)</param>

public Game(int player1, int player2)

{

this.GameFinished = 0;

this.BoardHigh = 20;

this.BoardWide = 20;

this.Player1 = player1;

this.Player2 = player2;

this.CountMapSets = 0;

Map = new int[BoardWide, BoardHigh];

for(int i = 0; i < BoardWide; i++)

{

for (int j = 0; j < BoardHigh; j++)

{

Map[i, j] = 0;

}

}

}

/// <summary>

/// Sets up the map by recieving string of 1 side each time.

/// </summary>

/// <param name="Givenmap"> the given map</param>

public void SetUpMap(string Givenmap)

{

Givenmap = Givenmap.Replace("\n", "");

for(int i = 0; i < BoardWide; i++)

{

for(int j = 0; j < BoardHigh; j++)

{

if(int.Parse(Givenmap.Substring(i + (20\*j),1)) == 1)

{

Map[j, i] = 1;

}

}

}

CountMapSets++;

}

public int Player1

{

get

{

return player1;

}

set

{

player1 = value;

}

}

public int Player2

{

get

{

return player2;

}

set

{

player2 = value;

}

}

public int CountMapSets1

{

get

{

return CountMapSets;

}

set

{

CountMapSets = value;

}

}

public int GameFinished1

{

get

{

return GameFinished;

}

set

{

GameFinished = value;

}

}

public int[,] GetMap()

{

return Map;

}

/// <summary>

/// Sets the map on the col and row to zero and checks if it hit something.

/// </summary>

/// <param name="col"> the column to hit</param>

/// <param name="row"> the row to hit</param>

/// <returns></returns>

public bool HitMap(int col, int row)

{

bool toret = false;

if(Map[col,row] == 1) // if there is a ship

{

toret = true;

}

Map[col, row] = 0;

return toret;

}

}

}

Code Printout – Game.cs – end

program.cs

קובץ זה הוא מחלקה הממוקמת בתוך תיקייה בשם סרבר (הנמוכה יותר), מחלקה זו אחראית על הרצת התוכנה ובדיקת תקינות קבצי המידע שבה (כל קבצי הטקסט) כך ששאר המחלקות יוכלו לרוץ באופן חלק, בנוסף לכך מחלקה זו כותבת לקובץ לוג כל פעם שהיא רצה.

Code Printout – Program.cs – start

using System;

using System.IO;

namespace Server

{

class Program

{

/// <summary>

/// This class stands for checking all the files that should to be are there, else it creates them

/// This class also stands for running the server.

/// </summary>

static void Main(string[] args)

{

DateTime now = DateTime.Now;

Server s = new Server(9999);

if (!File.Exists("chat.txt"))

{

File.CreateText("chat.txt");

}

if (!File.Exists("log.txt"))

{

File.CreateText("log.txt");

}

if (!File.Exists("database.txt"))

{

File.CreateText("database.txt");

}

Console.ForegroundColor = ConsoleColor.Green;

Console.Title = "Server";

Console.WriteLine("Server status: ON.\nDon't forget to reset the Config.txt on the Client's file to {0}\nWaiting for connection...", s.Ip);

using (StreamWriter sw = new StreamWriter("log.txt", true)) // Saves "towrite" on the log file.

{

sw.WriteLine("Restarts the server at: " + now.ToString() + "\nIP: " + s.Ip );

sw.Flush();

}

s.Start();

}

}

}

Code Printout – Program.cs – end

User.cs

קובץ זה הוא מחלקה הממוקמת בתוך תיקייה בשם סרבר (הנמוכה יותר), מחלקה זו אחראית על בניית משתמש לכל שחקן שמתחבר כך שהשרת יכול לשלוף את זה בכל רגע שיצטרך, בנוסף לכך הפונקציה אחראית על שינויים בתוך קובץ

בו היא משנה כמות ניצחונות הפסדים ורמת אדמין..Database.txt

Code Printout – User.cs – start

using System.IO;

namespace Server

{

class User

{

/// <summary>

/// This class working with the database file!

/// This class handles all the changes in the database file except of registering a new user.

/// This class changing and sending ranks, wins and loses for each user it opens ((if nessesary))

/// This class's objects are users from the database file.

/// </summary>

private const string FILENAME = "database.txt";

private string username;

private int wins;

private int loses;

private int rank;

private int ID;

private int firstcut;

private int secondcut;

private int usernameline;

private string[] CodeLines;

private string WLR;

/// <summary>

/// BUILDER - ejects the user's details and setting them up to be able to send easly.

/// </summary>

/// <param name="username"> the user username</param>

/// <param name="usernameline"> the user username line in the database.</param>

/// <param name="ID"> his defined ID.</param>

public User(string username, int usernameline, int ID)

{

this.usernameline = usernameline;

this.ID = ID;

this.username = username;

this.CodeLines = File.ReadAllLines("database.txt");

this.WLR = CodeLines[usernameline + 2];

this.firstcut = WLR.IndexOf("/");

this.secondcut = WLR.LastIndexOf("/") - 1;

this.rank = int.Parse(WLR.Substring(0, firstcut));// cuts the details from the database file.

this.wins = int.Parse(WLR.Substring(firstcut+1,secondcut - firstcut));// cuts the details from the database file.

this.loses = int.Parse(WLR.Substring(secondcut+2, WLR.Length - (2 + secondcut))); // cuts the details from the database file.

}

/// <summary>

/// Changing a line in the database file the receiving the new text and the line to edit.

/// </summary>

/// <param name="newText"> the new Text to set</param>

/// <param name="line\_to\_edit"> the line to set.</param>

public static void lineChanger(string newText, int line\_to\_edit)

{

string[] arrLine = File.ReadAllLines(FILENAME);

arrLine[line\_to\_edit - 1] = newText;

File.WriteAllLines(FILENAME, arrLine);

}

/// <summary>

/// this function adds a win to the user's details

/// </summary>

public void AddWin()

{

wins++;

string towrite = rank + "/" + wins + "/" + loses;

lineChanger(towrite, usernameline + 3);

}

/// <summary>

/// this function adds a lose to the user's details

/// </summary>

public void AddLose()

{

loses++;

string towrite = rank + "/" + wins + "/" + loses;

lineChanger(towrite, usernameline + 3);

}

/// <summary>

/// this function sets the wins to the user's details

/// </summary>

/// <param name="wins"> .</param>

public void SetWin(int wins)

{

this.wins = wins;

string towrite = rank + "/" + wins + "/" + loses;

lineChanger(towrite, usernameline + 3);

}

/// <summary>

/// this function sets the loses to the user's details

/// </summary>

/// <param name="loses">the loses ammount to set</param>

public void SetLose(int loses)

{

this.loses = loses;

string towrite = rank + "/" + wins + "/" + loses;

lineChanger(towrite, usernameline + 3);

}

/// <summary>

/// this function sets the rank to the user's details

/// </summary>

/// <param name="rank">the rank to set</param>

public void SetRank(int rank)

{

this.rank = rank;

string towrite = rank + "/" + wins + "/" + loses;

lineChanger(towrite, usernameline + 3);

}

/// <summary>

/// sends how many wins the user has

/// </summary>

/// <returns> the wins ammount</returns>

public int GetW()

{

return this.wins;

}

/// <summary>

/// sends how many loses the user has

/// </summary>

/// <returns>the lose ammount</returns>

public int GetL()

{

return this.loses;

}

/// <summary>

/// sends the rank of the user

/// </summary>

/// <returns>the rank</returns>

public int GetR()

{

return this.rank;

}

/// <summary>

/// sends the username of the user.

/// </summary>

/// <returns>the username</returns>

public string GetUN()

{

return this.username;

}

}

}

Code Printout – User.cs – end

Server.cs

קובץ זה הוא מחלקה הממוקמת בתוך תיקייה בשם סרבר (הנמוכה יותר), מחלקה זו אחראית על בניית השרת עצמו והוספת לקוחות עליו (מטיפוס קליינט הנדלר).

Code Printout – Server.cs – start

using System.Net;

using System.Net.Sockets;

namespace Server

{

class Server

{

/// <summary>

/// This class creates the server and sets it able to get more clients.

/// </summary>

private TcpListener listener;

private IPAddress ip;

private int port;

~Server()

{

listener.Stop();

}

public TcpListener Listener

{

get

{

return listener;

}

set

{

listener = value;

}

}

public IPAddress Ip

{

get

{

return ip;

}

set

{

ip = value;

}

}

public int Port

{

get

{

return port;

}

set

{

port = value;

}

}

/// <summary>

/// BUILDER - creates the server by the IP address of the computer and the given port.

/// </summary>

/// <param name="port"> the PORT number</param>

public Server(int port)

{

var host = Dns.GetHostName();

this.ip = IPAddress.Parse(Dns.GetHostByName(host).AddressList[0].ToString());

this.port = port;

this.listener = new TcpListener(ip, this.port);

}

/// <summary>

/// This function recieves clients and creates a Clienthandler to handle the client.

/// </summary>

public void Start()

{

listener.Start();

while (true)

{

var newClient = listener.AcceptTcpClient();

ClientHandler ch = new ClientHandler(newClient);

ClientHandler.clntList.Add(ch);

}

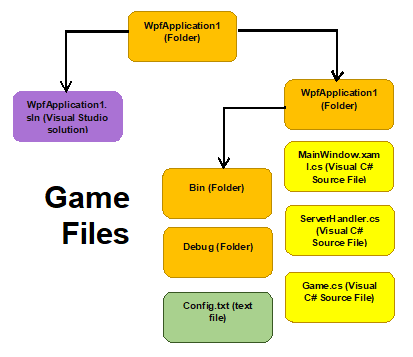
}

}

}

Code Printout – Server.cs – end

## קבצי המשחק



WpfApplication1.sln

הקובץ הזה הוא קובץ שנוצר על ידי וויזואל סטודיו ומשומש בשביל לכתוב את הפרויקט, הקובץ ממוקם בתוך תיקיית המשחק (העליונה יותר), הקובץ עצמו משתמש במחלקות בנמצאות בתוך תיקיית המשחק (הנמוכה יותר)

\*wpfapplication1 = תיקיית המשחק

MainWindow.xaml.cs

קובץ זה הוא מחלקה הממוקמת בתוך תיקיית המשחק (הנמוכה יותר), מחלקה זו אחראית על טיפול במסך הלקוח, משמע כל דבר שצריך להציג או לקבל על ידי לחיצה או הזנה של נתונים שלא כוללת את המשחק עצמו קורה במחלקה הזאת

Code Printout – MainWindow.xaml.cs – start

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.IO;

using System.Threading;

namespace WpfApplication1

{

/// <summary>

/// MAIN WINDOW: This window have 6 pages!

/// 1. LOG IN / REGISTER PAGE: This page takes care about the log in and register request - sends the log in / register details to the server for confirm

/// 2. Main Page: 3 optional choices - PLAY INFO EXIT: PLAY means you attempts to start a game. INFO means you want to se

/// </summary>

public partial class MainWindow : Window

{

public ServerHandler Client;

public Game game;

private int usernumber;

private int wins; // Player wins

private int loses;// Player's loses

private int rank;// Player's rank

public int playernumber;//first/second

public int enemyid;//server's ID

public string myusername;// player's username

public MainWindow()

{

InitializeComponent();

Client = new ServerHandler();

}

/// <summary>

/// updates stats into rank, wins and loses ints.

/// </summary>

public void GetStats()

{

Client.WriteThread("STS");

string recieved = Client.ReadThread();

int firstcut = recieved.IndexOf("/");

int secondcut = recieved.LastIndexOf("/") - 1;

this.rank = int.Parse(recieved.Substring(0, firstcut));

this.wins = int.Parse(recieved.Substring(firstcut + 1, secondcut - firstcut));

this.loses = int.Parse(recieved.Substring(secondcut + 2, recieved.Length - (2 + secondcut)));

}

/// <summary>

///Checking the username and password, if it's match to log.txt file it saves it on trueusername and sets the content in Check works, else sets failed.

/// </summary>

private void Continue\_Click( object sender, RoutedEventArgs e)

{

string givenusername = Username.Text;

string givenpassword = Password.Text;

string recievefromserver;

Client.WriteThread("LOG: \n" + givenusername + "\n" + givenpassword + "\n");

recievefromserver = Client.ReadThread();

if (recievefromserver.Contains("Logged in"))

{

usernumber = int.Parse(recievefromserver.Substring(15, recievefromserver.Length - 15));

myusername = givenusername;

Check.Content = "works";

ShowMainScreen();

}

else

{

Check.Content = "failed";

}

}

/// <summary>

/// Registers user by sending his name and password to the server and if there is no user with his username it sends "registered" but if there is it sends can't register.

/// </summary>

private void Register\_Click(object sender, RoutedEventArgs e)

{

string givenusername = Username.Text;

string givenpassword = Password.Text;

Client.WriteThread("REG: \n" + givenusername + "\n" + givenpassword + "\n");

if (Client.ReadThread() == "Registered")

{

Check.Content = "Registered";

myusername = givenusername;

ShowMainScreen();

}

else

{

Check.Content = "Can't Register";

}

}

/// <summary>

/// starts the info page by switching visibilities

/// </summary>

private void Info\_Click(object sender, RoutedEventArgs e)

{

HelpPage.Visibility = Visibility.Visible;

MainPage.Visibility = Visibility.Hidden;

}

/// <summary>

/// sends a quit message to the server and closes the stream with the server as it closes the application.

/// </summary>

private void Exit\_Click(object sender, RoutedEventArgs e)

{

Client.WriteThread("OUT");

Client.Close();

this.Close();// exit the game

}

/// <summary>

/// sends to the server a message that the client looks up for a player to play with.

/// </summary>

private void Play\_Click(object sender, RoutedEventArgs e)

{

PlayPage.Visibility = Visibility.Visible;

MainPage.Visibility = Visibility.Hidden;

Thread.Sleep(100);

// GetStats();// updates stats

Client.WriteThread("PLY");

// Statslabel.Content = "Stats: \n Rank: " + rank + "\n Wins: " + wins + "\n Loses: " + loses;

Task<bool> wait = new Task<bool>(Play\_Click\_Extended);

wait.Start();

while (!wait.Result)

{

}

this.Hide();

GetStats();// updates stats

Statslabel.Content = "Stats: \n Rank: " + rank + "\n Wins: " + wins + "\n Loses: " + loses;

try

{

game = new Game(Client, playernumber, myusername, enemyid); // turns to the game with the player number and enemy's client specific number

game.ShowDialog();//Starts the game and back here as the game finish

Thread.Sleep(500);

this.Show();

}

catch

{

game.Close();

this.Show();

MessageBox.Show("Error has been occured, this game willn't count. \nif you was about to win please contact with the admins by chatting them as soon as possible.");

}

}

/// <summary>

/// waiting for a GSN and when it finds one when someone else trying to connect to a game it sends a number to the client to update him if he starts or not.

/// </summary>

/// <returns> returns if the game is going to start</returns>

private bool Play\_Click\_Extended()

{

string recieved = Client.ReadThread();

bool found = false;

while (recieved.Substring(0,3) != "GSN")

{

recieved = Client.ReadThread();

}

found = true;

playernumber = int.Parse(recieved.Substring(3, 1)); //who starts: 1 starter, 2 play after him

enemyid = int.Parse(recieved.Substring(4, 1)); // enemy's Client Specific number.

return found;

}

/// <summary>

/// Exits to Main by a button click

/// </summary>

private void ExitToMain\_Click(object sender, RoutedEventArgs e)

{

ShowMainScreen();//The function which hides all the other grids and setting the Main one.

}

/// <summary>

/// Exits to Main by a button click on the Play screen and sets the game searching on false.

/// </summary>

private void ExitPLY\_Click(object sender, RoutedEventArgs e)

{

Client.WriteThread("LGF"); //Leave Game Finder

ShowMainScreen();//The function which hides all the other grids and setting the Main one.

}

/// <summary>

/// Hides all the screens and sets the main screen visible.

/// </summary>

private void ShowMainScreen()

{

if (Suggestion\_Page.IsVisible)

Suggestion\_Page.Visibility = Visibility.Hidden;

if (register.IsVisible)

register.Visibility = Visibility.Hidden;

if (HelpPage.IsVisible)

HelpPage.Visibility = Visibility.Hidden;

if (PlayPage.IsVisible)

PlayPage.Visibility = Visibility.Hidden;

MainPage.Visibility = Visibility.Visible;

}

/// <summary>

/// Closes the window and sends OUT to the server.

/// </summary>

private void IntroWindow\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

{

Client.WriteThread("OUT");

Client.Close();

}

private void Button\_Click(object sender, RoutedEventArgs e)

{

HelpPage.Visibility = Visibility.Hidden;

Suggestion\_Page.Visibility = Visibility.Visible;

}

/// <summary>

/// sends to the server what the client wrote with his name to set it on the chat.txt file.

/// </summary>

private void ChatButton\_Click(object sender, RoutedEventArgs e)

{

Client.WriteThread("CHT: " + myusername + ":\n" + ChatBox.Text);

ShowMainScreen();

ChatBox.Text = null; // deletes what the client wrote before.

}

}

}

Code Printout – MainWindow.xaml.cs – end

ServerHandler.cs

קובץ זה הוא מחלקה הממוקמת בתוך תיקיית המשחק (הנמוכה יותר), מחלקה זו אחראית העברת מסרים בין הלקוח לשרת, מחלקה זו היא בעצם מקשרת בין השרת ללקוח בצורה של העברה וקבלה של מחרוזות

Code Printout – ServerHandler.cs – start

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Net;

using System.Net.Sockets;

using System.Threading;

using System.IO;

namespace WpfApplication1

{

/// <summary>

/// This class handles the transmission between the client and the server.

/// </summary>

public class ServerHandler

{

public static TcpClient c = new TcpClient();

/// <summary>

/// creates the server by taking Port and IP from Config.txt file.

/// </summary>

public ServerHandler()

{

string IP;

int ServerPort = 9999;

if (File.Exists("Config.txt"))

{

IP = File.ReadAllText("Config.txt");

}

else

{

IP = "127.0.0.1";

}

c.Connect(IPAddress.Parse(IP), ServerPort);

}

/// <summary>

///reads from the server

/// </summary>

/// <returns> returns the message which recieved from the server</returns>

public string ReadThread()

{

byte[] data = new byte[1024];

int len = c.GetStream().Read(data, 0, data.Length);

string msg = Encoding.UTF8.GetString(data, 0, len);

return msg;

}

/// <summary>

/// // writes to the server

/// </summary>

/// <param name="msg"> the message it sends to the server.</param>

public void WriteThread(string msg)

{

byte[] data = Encoding.UTF8.GetBytes(msg);

c.GetStream().Write(data, 0, data.Length);

c.GetStream().Flush();

}

/// <summary>

/// Closes the transmission with the server.

/// </summary>

public void Close()

{

//close socket

c.Close();

}

}

}

Code Printout – ServerHanlder.cs – end

Game.xaml.cs

קובץ זה הוא מחלקה הממוקמת בתוך תיקיית המשחק (הנמוכה יותר), מחלקה זו אחראית על תצוגת המשחק עצמו, בעזרת הפונקציה המקשרת בין הלקוח לשרת הפונקציה הזו מקבלת ומעבירה מידע ובכך יודעת מה להציג על המסך בזמן המשחק.

Code Printout – Game.xaml.cs – start

using System;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace WpfApplication1

{

/// <summary>

///Game Window: This window is responsable of all the game matters

///First setting up the ships and sends it to the server

///Second Hitting the enemy on the chosen locations and sends it to the server

///third Sending a win message to the server;

/// </summary>

public partial class Game : Window

{

bool gameover = false;

int Tocatch;

int x = -1;

int y = -1;

Rectangle rec = new Rectangle();

Image[] xlimgs = new Image[2];// placeable extra large ships = 2

Image[] limgs = new Image[3];//placeable large ships = 3

Image[] mimgs = new Image[4];// placeable medium ships = 4

Image[] simgs = new Image[3];// placeable medium ships = 3

ServerHandler Client;

int playernumber;

int enemyid;

bool Stopwaitforenemy;

string enemyname;

int textblockcounter;//counts how many lines were written to purge it

int[,] field; // 0 means empty, 1 means full, 2 means hit, 3 means shooted in an empty space

bool preparing = true;

/// <summary>

/// This function creates a game

/// </summary>

/// <param name="Client"> Stands for the transmission between the server and the client</param>

/// <param name="playernumber"> The ID of the client</param>

/// <param name="myusername"> The username of my client</param>

/// <param name="enemyid"> The enemy ID number</param>

public Game(ServerHandler Client, int playernumber,string myusername, int enemyid)

{

this.textblockcounter = 0;

this.rec.Fill = Brushes.Red;

this.rec.Opacity = 0.7;

InitializeComponent();

this.Client = Client;

this.playernumber = playernumber;

this.enemyid = enemyid;

Client.WriteThread("GEN"+enemyid);//Get Enemy Name

enemyname = Client.ReadThread();

field = new int[20, 20];

for (int i = 0; i < field.GetLength(0); i++)

{

for (int j = 0; j < field.GetLength(1); j++)

{

field[i, j] = 0; //Sets an empty field

}

}

if(playernumber == 1) //writes on the upper labels the names of the players by their player number.

{

Player1lbl.Content = Player1lbl.Content+ myusername;

Player2lbl.Content = Player2lbl.Content + enemyname;

}

else

{

Player1lbl.Content = Player1lbl.Content + enemyname;

Player2lbl.Content = Player2lbl.Content + myusername;

}

this.Tocatch = xlimgs.Length\*4 + limgs.Length\*3 + mimgs.Length\*2 + simgs.Length;

}

/// <summary>

/// This function returns the mouse location

/// </summary>

/// <returns>returns the mouse location</returns>

private int[] GetMouseLoc()

{

var point = Mouse.GetPosition(board);

int row = 0;

int col = 0;

double accumulatedHeight = 0.0;

double accumulatedWidth = 0.0;

// calc row mouse was over

foreach (var rowDefinition in board.RowDefinitions)

{

accumulatedHeight += rowDefinition.ActualHeight;

if (accumulatedHeight >= point.Y)

break;

row++;

}

// calc col mouse was over

foreach (var columnDefinition in board.ColumnDefinitions)

{

accumulatedWidth += columnDefinition.ActualWidth;

if (accumulatedWidth >= point.X)

break;

col++;

}

int[] tosend = new int[2];

tosend[0] = col;

tosend[1] = row;

return tosend;

}

/// <summary>

/// Event: Catches the mouse movement of the grid and sets the red rectangle if needed.

/// </summary>

private void Mouse\_Move(object sender, MouseEventArgs e)

{

int[] getloc = GetMouseLoc();

int col = getloc[0];

int row = getloc[1];

if (!(playernumber == 1 && col < 10 || playernumber == 2 && col >= 10 ) &&(field[col,row] == 0))// checks if the placement is in the zone

{

board.Children.Remove(rec);

Grid.SetColumn(rec, col);

Grid.SetRow(rec, row);

board.Children.Add(rec);

}

else//if it's inbound

{

if (board.Children.Contains(rec))

board.Children.Remove(rec);

}

}

/// <summary>

/// This event catch a press on the left mouse button.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Board\_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

int[] getloc = GetMouseLoc();

int col = getloc[0]; // sets last col to the current one

int row = getloc[1];// sets last row to the current one

// X and Y presents the cordinates of the 1 before the last tap location.

// Col and Row presents the cordinates of tapped location.

if (preparing)

{

if (playernumber == 1 && col < 10 || playernumber == 2 && col >= 10)// checks if the placement is in the zone

{

if (x != -1 && y != -1) //means last tap was to place or hadn't tapped before.

{

if (field[x, y] == 0)

{

if (x == col - 3 && y == row)

{

CreateExtraLargeSubFlat(x, y);

}

if (x - 3 == col && y == row)

{

CreateExtraLargeSubFlat(col, y);

}

if (x == col - 2 && y == row)

{

CreateLargeSubFlat(x, y);

}

if (x - 2 == col && y == row)

{

CreateLargeSubFlat(col, y);

}

if (x == col - 1 && y == row)

{

CreateMediumSubFlat(x, y);

}

if (x - 1 == col && y == row)

{

CreateMediumSubFlat(col, y);

}

if (x == col && y == row)

{

CreateSmallSub(col, row);

}

x = -1;

y = -1;

}

else//pressed on a battleship

{

RemoveShip(x, y);

x = -1;

y = -1;

}

}

else // didn't tapped before

{

x = col;

y = row;

}

}

else //incase it's out of range

{

Writeintotextblock("Out of range");

}

if (UpdateShipsLeft()) // updates the "shipsleft" label and enters if there are no ships left

{

Writeintotextblock("No ships left, Press ENTER to start playing");

}

}// all of that happens if the players are still preparing their ships for the game.

else if ((playernumber == 2 && col < 10 || playernumber == 1 && col >= 10) && (field[col, row] == 0))

{

if (Stopwaitforenemy) // Check if it's his turn.

{

Client.WriteThread("HIT: " + col + "," + row);

string recieved = Client.ReadThread();

Rectangle placerec = new Rectangle();

if (recieved == "C") // incase caught

{

field[col, row] = 2;

Writeintotextblock("Cought him!");

Tocatch--; // if Tocatch == 0 he won

placerec.Opacity = 0.8;

if (Tocatch == 0) // checks if there are no ships left to catch to send a win to the server.

{

Client.WriteThread("WIN");// updates win

gameover = true;

Finish.Visibility = Visibility.Visible;

textblock.Text = "You WON! Press enter to continue";

}

}

if (recieved == "M") // incase missed

{

field[col, row] = 3;

Writeintotextblock("Missed...");

placerec.Opacity = 0.4;

}

Grid.SetColumn(placerec, col); // Sets the column of the object

Grid.SetRow(placerec, row);// Sets the row of the object

placerec.Fill = Brushes.Black;

board.Children.Add(placerec); // adds it to the board.

Shipsleft.Content = "Ships to catch: " + Tocatch.ToString();

if (Tocatch == 0)

{

Client.WriteThread("WIN");// re-check if there is any win.

gameover = true;

Finish.Visibility = Visibility.Visible;

textblock.Text = "You WON! Press enter to continue";

}

Stopwaitforenemy = false;

Waiting1();

}

else

{

Writeintotextblock("Wait till your enemy finish his turn");

}

}

else if ((playernumber == 2 && col < 10 || playernumber == 1 && col >= 10) && (field[col, row] == 2))

{

Writeintotextblock("The ship at: " + col + "," + row + " already hitted"); // incase you shoot a ship that you already shoot.

}

else if ((playernumber == 2 && col < 10 || playernumber == 1 && col >= 10) && (field[col, row] == 3))

{

Writeintotextblock("You already tried to hit: " + col + "," + row + " and missed..."); // incase you try to shoot where you already shoot.

}

}

private void CreateExtraLargeSubFlat(int x, int y) // Creates extra large ship and sets it into the field, board and array.

{

bool full = true;

bool placeable = false;

Image img = new Image();

for (int i = 0; i < xlimgs.Length && !placeable; i++)// checks if there are large ships left

{

if (xlimgs[i] == null)

{

if (field[x, y] == 0 && field[x + 1, y] == 0 && field[x + 2, y] == 0 && field[x + 3, y] == 0)

{

xlimgs[i] = img;

placeable = true;

}

full = false;

}

}

if (placeable)

{

board.Children.Add(img);

if (x + 3 < 10) //first half x = loc 3 = additational space 10 = half board

img.Source = new BitmapImage(new Uri("pack://application:,,,/Images/submarine4Left.png"));

else

img.Source = new BitmapImage(new Uri("pack://application:,,,/Images/submarine4Right.png"));

Grid.SetColumn(img, x);

Grid.SetRow(img, y);

Grid.SetColumnSpan(img, 4);

field[x, y] = 1; // fills the array

field[x + 1, y] = 1;

field[x + 2, y] = 1;

field[x + 3, y] = 1;

}

else

{

if (field[x, y] == 1 || field[x + 1, y] == 1 || field[x + 2, y] == 1 || field[x + 3, y] == 1)

{

Writeintotextblock("You can't place a ship on another one.");

}

if (full)

{

Writeintotextblock("No more ships left!");

}

if (!full || !(field[x, y] == 1 || field[x + 1, y] == 1 || field[x + 2, y] == 1 || field[x + 3, y] == 1))

{

//Any other reason.

}

}

}

/// <summary>

/// Creates large ship and sets it into the field, board and array.

/// </summary>

/// <param name="x"> cordinate</param>

/// <param name="y">cordinate</param>

private void CreateLargeSubFlat(int x, int y)

{

bool full = true;

bool placeable = false;

Image img = new Image();

for (int i = 0; i < limgs.Length && !placeable; i++)// checks if there are large ships left

{

if (limgs[i] == null)

{

if (field[x, y] == 0 && field[x + 1, y] == 0 && field[x + 2, y] == 0)

{

limgs[i] = img;

placeable = true;

}

full = false;

}

}

if (placeable)

{

board.Children.Add(img);

if (x + 2 < 10) //first half x = loc 3 = additational space 10 = half board

img.Source = new BitmapImage(new Uri("pack://application:,,,/Images/submarine3Left.png"));

else

img.Source = new BitmapImage(new Uri("pack://application:,,,/Images/submarine3Right.png"));

Grid.SetColumn(img, x);

Grid.SetRow(img, y);

Grid.SetColumnSpan(img, 3);

field[x, y] = 1; // fills the array

field[x + 1, y] = 1;

field[x + 2, y] = 1;

}

else

{

if (field[x, y] == 1 || field[x + 1, y] == 1 || field[x + 2, y] == 1)

{

Writeintotextblock("You can't place a ship on another one.");

}

if (full)

{

Writeintotextblock("No more ships left!");

}

if(!full || !(field[x, y] == 1 || field[x + 1, y] == 1 || field[x + 2, y] == 1))

{

//Any other reason.

}

}

}

/// <summary>

/// Creates medium ship and sets it into the field, board and array.

/// </summary>

/// <param name="x"> cordinate</param>

/// <param name="y">cordinate</param>

private void CreateMediumSubFlat(int x, int y)

{

bool full = true;

bool placeable = false;

Image img = new Image();

for (int i = 0; i < mimgs.Length && !placeable; i++)// Checks if there are medium ships left

{

if (mimgs[i] == null)

{

if (field[x, y] == 0 && field[x + 1, y] == 0)

{

mimgs[i] = img;

placeable = true;

}

}

full = false;

}

}

if (placeable)

{

board.Children.Add(img);

if(x + 1 < 10) //first half x = loc 1 = additational space 10 = half board

img.Source = new BitmapImage(new Uri("pack://application:,,,/Images/submarine2Left.png"));

else

img.Source = new BitmapImage(new Uri("pack://application:,,,/Images/submarine2Right.png"));

Grid.SetColumn(img, x);

Grid.SetRow(img, y);

Grid.SetColumnSpan(img, 2);

field[x, y] = 1; // fills the array

field[x + 1, y] = 1;

}

else

{

if (field[x, y] == 1 || field[x + 1, y] == 1)

{

Writeintotextblock("You can't place a ship on another one.");

}

if (full)

{

Writeintotextblock("No more ships left!");

}

if (!full || !(field[x, y] == 1 || field[x + 1, y] == 1))

{

//Any other reason.

}

}

}

/// <summary>

/// Creates small ship and sets it into the field, board and array.

/// </summary>

/// <param name="x"> cordinate</param>

/// <param name="y">cordinate</param>

private void CreateSmallSub(int x, int y)

{

bool full = true;

bool placeable = false;

Image img = new Image();

for (int i = 0; i < simgs.Length && !placeable; i++)// Checks if there are small ships left

{

if (simgs[i] == null)

{

if (field[x, y] == 0)

{

simgs[i] = img;

placeable = true;

}

full = false;

}

}

if (placeable)

{

board.Children.Add(img);

if (x < 10) //first half x = loc 2 = additational space 10 = half board

img.Source = new BitmapImage(new Uri("pack://application:,,,/Images/submarine1Left.png"));

else

img.Source = new BitmapImage(new Uri("pack://application:,,,/Images/submarine1Right.png"));

Grid.SetColumn(img, x);

Grid.SetRow(img, y);

field[x, y] = 1; // fills the array

}

else

{

if (field[x, y] == 1)

{

Writeintotextblock("You can't place a ship on another one.");

}

if (full)

{

Writeintotextblock("No more ships left!");

}

if (!full || !(field[x, y] == 1))

{

//Any other reason.

}

}

}

/// <summary>

///Removes ship from it's array, remove it from the field and board childrens ///

///</summary>

/// <param name="x"> cordinate</param>

/// <param name="y">cordinate</param>

private void RemoveShip(int x, int y)

{

bool foundship = false;

for (int i = 0; i < simgs.Length && !foundship; i++)

if (simgs[i] != null)

{

if (Grid.GetColumn(simgs[i]) == x && Grid.GetRow(simgs[i]) == y)

{

board.Children.Remove(simgs[i]);

simgs[i] = null;

field[x, y] = 0;

foundship = true;

}

}

for (int i = 0; i < mimgs.Length && !foundship; i++)

if (mimgs[i] != null)

{

if ((Grid.GetColumn(mimgs[i]) == x) && (Grid.GetRow(mimgs[i]) == y))

{

board.Children.Remove(mimgs[i]);

mimgs[i] = null;

field[x, y] = 0;

field[x + 1, y] = 0;

foundship = true;

}

else if ((Grid.GetColumn(mimgs[i]) == x - 1) && (Grid.GetRow(mimgs[i]) == y))

{

board.Children.Remove(mimgs[i]);

mimgs[i] = null;

field[x - 1, y] = 0;

field[x, y] = 0;

foundship = true;

}

}

for (int i = 0; i < limgs.Length && !foundship; i++)

if (limgs[i] != null)

{

if ((Grid.GetColumn(limgs[i]) == x) && (Grid.GetRow(limgs[i]) == y))

{

board.Children.Remove(limgs[i]);

limgs[i] = null;

field[x, y] = 0;

field[x + 1, y] = 0;

field[x + 2, y] = 0;

foundship = true;

}

else if ((Grid.GetColumn(limgs[i]) == x - 1) && (Grid.GetRow(limgs[i]) == y))

{

board.Children.Remove(limgs[i]);

limgs[i] = null;

field[x - 1, y] = 0;

field[x, y] = 0;

field[x + 1, y] = 0;

foundship = true;

}

else if ((Grid.GetColumn(limgs[i]) == x - 2) && (Grid.GetRow(limgs[i]) == y))

{

board.Children.Remove(limgs[i]);

limgs[i] = null;

field[x - 2, y] = 0;

field[x - 1, y] = 0;

field[x, y] = 0;

foundship = true;

}

}

for (int i = 0; i < xlimgs.Length && !foundship; i++)

if (xlimgs[i] != null)

{

if ((Grid.GetColumn(xlimgs[i]) == x) && (Grid.GetRow(xlimgs[i]) == y))

{

board.Children.Remove(xlimgs[i]);

xlimgs[i] = null;

field[x, y] = 0;

field[x + 1, y] = 0;

field[x + 2, y] = 0;

field[x + 3, y] = 0;

foundship = true;

}

else if ((Grid.GetColumn(xlimgs[i]) == x - 1) && (Grid.GetRow(xlimgs[i]) == y))

{

board.Children.Remove(xlimgs[i]);

xlimgs[i] = null;

field[x - 1, y] = 0;

field[x, y] = 0;

field[x + 1, y] = 0;

field[x + 2, y] = 0;

foundship = true;

}

else if ((Grid.GetColumn(xlimgs[i]) == x - 2) && (Grid.GetRow(xlimgs[i]) == y))

{

board.Children.Remove(xlimgs[i]);

xlimgs[i] = null;

field[x - 2, y] = 0;

field[x - 1, y] = 0;

field[x, y] = 0;

field[x + 1, y] = 0;

foundship = true;

}

else if ((Grid.GetColumn(xlimgs[i]) == x - 3) && (Grid.GetRow(xlimgs[i]) == y))

{

board.Children.Remove(xlimgs[i]);

xlimgs[i] = null;

field[x - 3, y] = 0;

field[x - 2, y] = 0;

field[x - 1, y] = 0;

field[x, y] = 0;

foundship = true;

}

}

if(foundship)

{

Writeintotextblock("Ship Removed");

}

} /// <summary>

/// This function writes to textblock.

/// </summary>

/// <param name="thingtowrite"> the string to write.</param>

private void Writeintotextblock(string thingtowrite)

{

if (!gameover) // checks if the game is still on, in target to avoid writing after game ends.

{

if (textblockcounter >= 18)// max lines

{

textblock.Text = thingtowrite + "\n";

textblockcounter = 0;

}

else

{

textblock.Text += thingtowrite + "\n";

}

textblockcounter++;

}

}

/// <summary>

/// //Updates how many ships left and sets it into "Shipleft" label and sends true if no ships left

/// </summary>

/// <returns> returns true if there are no ships left..</returns>

private bool UpdateShipsLeft()

{

int extralarge = 0;

int large = 0;

int medium = 0;

int small = 0;

for (int i = 0; i < xlimgs.Length; i++)

if (xlimgs[i] == null)

extralarge++;

for (int i = 0; i < limgs.Length; i++)

if (limgs[i] == null)

large++;

for (int i = 0; i < mimgs.Length; i++)

if (mimgs[i] == null)

medium++;

for (int i = 0; i < simgs.Length; i++)

if (simgs[i] == null)

small++;

Shipsleft.Content = "extra large ships left: " + extralarge + "\n" + "large ships left: " + large + "\n" + "medium ships left: " + medium + "\n" + "small ships left: " + small + "\n";

if (small + medium + large + extralarge == 0)

return true;

return false;

}

/// <summary>

/// //Updates how many ships left and sets it into "Shipleft" label and sends true if no ships left

/// </summary>

/// <returns> returns how many ships left.</returns>

private int GetHowManyShipsLeft()

{

int extralarge = 0;

int large = 0;

int medium = 0;

int small = 0;

for (int i = 0; i < xlimgs.Length; i++)

if (xlimgs[i] == null)

extralarge++;

for (int i = 0; i < limgs.Length; i++)

if (limgs[i] == null)

large++;

for (int i = 0; i < mimgs.Length; i++)

if (mimgs[i] == null)

medium++;

for (int i = 0; i < simgs.Length; i++)

if (simgs[i] == null)

small++;

return small + medium + large + extralarge;

}

private string Waiting()

{

string recieved = Client.ReadThread();

if(recieved.Length>12)

if (recieved.Substring(0,12)== "Stop waiting")

return recieved.Remove(0,14);

return "failed";

}

private async void Waiting1()

{

string c;

Task<string> waiting = new Task<string>(Waiting);

waiting.Start();

c = await waiting;

if (c != "failed")

{

Client.WriteThread("CHW"); // Game Protocol to check if there is a winner.

if (Client.ReadThread() == "Enemy won")

{

gameover = true;

Finish.Visibility = Visibility.Visible;

textblock.Text = "Your enemy won, please press enter to continue..";

int col = int.Parse(c.Substring(0, c.IndexOf(",")));

int row = int.Parse(c.Substring(c.IndexOf(",") + 1));

if (col >= 0 && row >= 0)

{

Rectangle placerec = new Rectangle();

placerec.Opacity = 0.6;

Grid.SetColumn(placerec, col); // Sets the column of the object

Grid.SetRow(placerec, row);// Sets the row of the object

placerec.Fill = Brushes.Red;

board.Children.Add(placerec); // adds it to the board.

}

}

else

{

Stopwaitforenemy = true;

Writeintotextblock(" It's your turn to play");

int col = int.Parse(c.Substring(0, c.IndexOf(",")));

int row = int.Parse(c.Substring(c.IndexOf(",") + 1));

if (col >= 0 && row >= 0)

{

Rectangle placerec = new Rectangle();

placerec.Opacity = 0.6;

Grid.SetColumn(placerec, col); // Sets the column of the object

Grid.SetRow(placerec, row);// Sets the row of the object

placerec.Fill = Brushes.Red;

board.Children.Add(placerec); // adds it to the board.

}

}

}

}

/// <summary>

/// // checks if the game isn't over, and send Stop preparing and waiting for the other player

/// </summary>

private void Finish\_Click(object sender, RoutedEventArgs e)

{

if (!gameover)

{

if (GetHowManyShipsLeft() == 0)

{

Client.WriteThread("SPP" + enemyid);

Finishcontinue();

}

else

Writeintotextblock("You need to place more ships!");

}

else

{

this.Close();

}

}

/// <summary>

/// //async function that recognizes when player stop preparing

/// </summary>

private async void Finishcontinue()

{

preparing = false;

Finish.Visibility = Visibility.Hidden;

textblock.Text = "Waiting for the second player to finish his preparing, please wait";

Shipsleft.Content = "";

bool c;

Task<bool> waiting = new Task<bool>(Finish2);

waiting.Start();

c = await waiting;

if (c)

{

textblock.Text = "Start Playing";

SendField();

Waiting1();

}

}

private bool Finish2()

{

if (Client.ReadThread() == "Start Playing")

return true;

return false;

}

private void SendField()

{

string tosendfield = "FLD:\n";

for (int i = 0; i < field.GetLength(1); i++)

{

for (int j = 0; j < field.GetLength(0); j++)

{

tosendfield += field[i,j];

}

tosendfield += "\n";

}

Client.WriteThread(tosendfield);

}

}

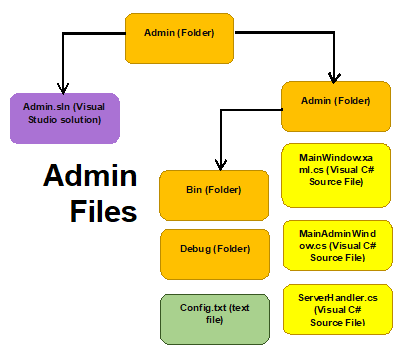
}

Code Printout – Game.xaml.cs – end

\*\*\* Config.txt

קובץ זה אחראי על האייפי שאליו הלקוח מנסה להתחבר, בכדי שיוכל להתחבר עליו להזין את האייפי שהשרת יושב עליו, מכוון שזה אינו קוד ומשתנה מעת לעת, אין סיבה להראות כאן את מה שנמצא בו. (אותו דבר גם על קבצי המידע בשרת)

## קבצי מערכת המנהלים



Admin.sln

הקובץ הזה הוא קובץ שנוצר על ידי וויזואל סטודיו ומשומש בשביל לכתוב את הפרויקט, הקובץ ממוקם בתוך תיקיית האדמין (העליונה יותר), הקובץ עצמו משתמש במחלקות בנמצאות בתוך תיקיית האדמין (הנמוכה יותר)

MainWindow.xaml.cs

קובץ זה הוא מחלקה הממוקמת בתוך תיקיית המשחק (הנמוכה יותר), מחלקה זו אחראית על טיפול במסך הלקוח בשלב הכניסה

Code Printout – MainWindow.xaml.cs – start

using System.Windows;

namespace Admin

{

/// <summary>

/// Log in Page - creating a connection between the client to the server by creating new ServerHandler

/// Also sends a log in requests to the server in target to get an okay to pass over to the next window ((MainAdminWindow)).

/// </summary>

public partial class MainWindow : Window

{

public MainAdminWindow AdminWindow;

public ServerHandler Client;

public int usernumber;

public string myusername;

/// <summary>

/// Builder - creates a connection with the server.

/// </summary>

public MainWindow()

{

InitializeComponent();

Client = new ServerHandler();

}

/// <summary>

/// Checking the username and password, if it's match to log.txt file it saves it on trueusername and sets the content in Check works, else sets failed.

/// </summary>

private void Continue\_Click(object sender, RoutedEventArgs e)

{

string givenusername = Username.Text; // recieves from the client

string givenpassword = Password.Text; // recieves from the client

string recievefromserver;

Client.WriteThread("LAG: \n" + givenusername + "\n" + givenpassword + "\n"); // sends an admin log in request to the server by the form it must to be.

recievefromserver = Client.ReadThread();

if (recievefromserver.Contains("Logged in")) // if the server accepts the request.

{

usernumber = int.Parse(recievefromserver.Substring(15, recievefromserver.Length - 15));

myusername = givenusername;

Check.Content = "works";

AdminWindow = new MainAdminWindow(Client, myusername, usernumber);

this.Visibility = Visibility.Hidden;

AdminWindow.ShowDialog();

this.Close();

}

else // incase the server deny the client.

{

Check.Content = "failed";

}

}

private void IntroWindow\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

{

Client.WriteThread("OUT");

Client.Close();

}

}

}

Code Printout – MainWindow.xaml.cs – end

MainAdminWindow.xaml.cs

קובץ זה הוא מחלקה הממוקמת בתוך תיקיית המשחק (הנמוכה יותר), מחלקה זו אחראית על טיפול במסך הלקוח כאשר הלקוח מאומת כאדמין ומבצע פעולות אדמין כמו צפייה בצאט או שינוי סיסמא למשתמש

Code Printout – MainAdminWindow.xaml.cs – start

using System.Windows;

namespace Admin

{

/// <summary>

/// Admin main window - that window deals with everything admin can do with his abilities - includes messing with the database,

/// chat and log files which are located in the server's floder.

/// </summary>

public partial class MainAdminWindow : Window

{

ServerHandler Client;

string username;

int usernumber;

string FindPlayerusername;

/// <summary>

/// BUILDER - Recieves the username of the admin, the client's connection with the server and his usernumber.

/// </summary>

/// <param name="Client"> The transmission with the server</param>

/// <param name="username"> The username of the client</param>

/// <param name="usernumber"> The ID.</param>

public MainAdminWindow(ServerHandler Client,string username,int usernumber)

{

InitializeComponent();

this.Client = Client;

this.username = username;

this.usernumber = usernumber;

}

/// <summary>

/// Sets up the Find Player screen after being pressed.

/// </summary>

private void Find\_Player\_Click(object sender, RoutedEventArgs e)

{

FindPlayerGrid.Visibility = Visibility.Visible;

InsideFindPlayerGrid.Visibility = Visibility.Hidden;

Edit\_DatabaseGrid.Visibility = Visibility.Hidden;

Show\_LogGrid.Visibility = Visibility.Hidden;

Show\_ChatGrid.Visibility = Visibility.Hidden;

}

/// <summary>

/// Start the search on the username and shows the results.

/// </summary>

private void FindPlayerContinue\_Click(object sender, RoutedEventArgs e)

{

Client.WriteThread("AFP: \n" + Username.Text);// PROTOCOL = Admin Find Player - checks if there is a username with the additational username.

string recieved = Client.ReadThread();

if (recieved.Contains("Couldn't find ")) // Incase he couldn't find the username.

{

InsideFindPlayerGrid.Visibility = Visibility.Hidden; // Removes (incase it's on) the detials grid.

Status.Content = recieved; // Writes that the system couldn't find the user

}

else

{

FindPlayerusername = Username.Text;

int rank = int.Parse(recieved.Substring(0, recieved.IndexOf("/")));

int wins = int.Parse(recieved.Substring(recieved.IndexOf("/") +1, recieved.LastIndexOf("/") -(recieved.IndexOf("/") + 1)));

int loses = int.Parse(recieved.Substring(recieved.LastIndexOf("/")+1));

ShowPlayerDetails(rank, wins, loses); // cuts the details which recieved from the server

}

}

/// <summary>

///

/// </summary>

/// <param name="rank"> the admin level of the client<</param>

/// <param name="wins">the wins ammount of the client</param>

/// <param name="loses"> the loses ammount of the client</param>

public void ShowPlayerDetails(int rank, int wins, int loses)

{

InsideFindPlayerGrid.Visibility = Visibility.Visible;

TextBlockRank.Text = rank.ToString();

TextBlockWins.Text = wins.ToString();

TextBlockLoses.Text = loses.ToString();

}

/// <summary>

/// attempts to save the changes that the client done.

/// </summary>

private void FindPlayerSaveChanges\_Click(object sender, RoutedEventArgs e)

{

try

{

Client.WriteThread("SET: \n"+ FindPlayerusername+"\n" +TextBoxRank.Text+"/"+TextBoxWins.Text+"/"+TextBoxLoses.Text);

SaveStatus.Content = Client.ReadThread();

TextBoxLoses.Text = "";

TextBoxRank.Text = "";

TextBoxWins.Text = "";

}

catch

{

SaveStatus.Content = "Failed to save";

}

}

/// <summary>

/// Recieves the database from the server and shows it.

/// </summary>

private void Edit\_DataBase\_Click(object sender, RoutedEventArgs e)

{

InsideFindPlayerGrid.Visibility = Visibility.Hidden;

FindPlayerGrid.Visibility = Visibility.Hidden;

Show\_LogGrid.Visibility = Visibility.Hidden;

Show\_ChatGrid.Visibility = Visibility.Hidden;

Edit\_DatabaseGrid.Visibility = Visibility.Visible;

Client.WriteThread("GDB: "); // Admin Get DataBase.

string recieved = Client.ReadThread(); // Recieves the database.

DatabaseTextBox.Text = recieved;// sets the database into database's Text Box.

}

/// <summary>

/// Sends the server the changes

/// </summary>

private void EditDatabaseButton\_Click(object sender, RoutedEventArgs e)

{

Client.WriteThread("EDB: \n" + DatabaseTextBox.Text); // Sends the new database to the server.

}

/// <summary>

/// Recieves the log from the sever and shows it.

/// </summary>

private void Show\_Log\_Click(object sender, RoutedEventArgs e)

{

FindPlayerGrid.Visibility = Visibility.Hidden;

InsideFindPlayerGrid.Visibility = Visibility.Hidden;

Edit\_DatabaseGrid.Visibility = Visibility.Hidden;

Show\_LogGrid.Visibility = Visibility.Visible;

Show\_ChatGrid.Visibility = Visibility.Hidden;

Client.WriteThread("GLG: ");// Admin command Get Log

string recieved = Client.ReadThread(); // recieves the Log.

LogTextBox.Text = recieved;// sets the log into log's Text Box.

}

/// <summary>

/// Recieves the chat from the sever and shows it.

/// </summary>

private void Clients\_Chat\_Click(object sender, RoutedEventArgs e)

{

FindPlayerGrid.Visibility = Visibility.Hidden;

InsideFindPlayerGrid.Visibility = Visibility.Hidden;

Edit\_DatabaseGrid.Visibility = Visibility.Hidden;

Show\_LogGrid.Visibility = Visibility.Hidden;

Show\_ChatGrid.Visibility = Visibility.Visible;

Client.WriteThread("GCC: ");// Admin command Get Client Chat

string recieved= Client.ReadThread(); //Gets the Chat from the server.

ChatBox.Text = recieved; // Sets the Chat that recieved from the server in the text box.

}

/// <summary>

/// Deletes the chat in the server.

/// </summary>

private void DeleteClientChatButton\_Click(object sender, RoutedEventArgs e)

{

Client.WriteThread("DCC: " + username); //Admin Command, Delete Client Chat.

ChatBox.Text = "Client Chat Deleted, Press on Client Chat Button for re-check.";

}

}

}

Code Printout – MainAdminWindow.xaml.cs – end

ServerHandler.cs

קובץ זה הוא מחלקה הממוקמת בתוך תיקיית המשחק (הנמוכה יותר), מחלקה זו אחראית העברת מסרים בין הלקוח לשרת, מחלקה זו היא בעצם מקשרת בין השרת ללקוח בצורה של העברה וקבלה של מחרוזות

Code Printout – ServerHandler.cs – start

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Net;

using System.Net.Sockets;

using System.Threading;

using System.IO;

namespace WpfApplication1

{

/// <summary>

/// This class handles the transmission between the client and the server.

/// </summary>

public class ServerHandler

{

public static TcpClient c = new TcpClient();

/// <summary>

/// creates the server by taking Port and IP from Config.txt file.

/// </summary>

public ServerHandler()

{

string IP;

int ServerPort = 9999;

if (File.Exists("Config.txt"))

{

IP = File.ReadAllText("Config.txt");

}

else

{

IP = "127.0.0.1";

}

c.Connect(IPAddress.Parse(IP), ServerPort);

}

/// <summary>

///reads from the server

/// </summary>

/// <returns> returns the message which recieved from the server</returns>

public string ReadThread()

{

byte[] data = new byte[1024];

int len = c.GetStream().Read(data, 0, data.Length);

string msg = Encoding.UTF8.GetString(data, 0, len);

return msg;

}

/// <summary>

/// // writes to the server

/// </summary>

/// <param name="msg"> the message it sends to the server.</param>

public void WriteThread(string msg)

{

byte[] data = Encoding.UTF8.GetBytes(msg);

c.GetStream().Write(data, 0, data.Length);

c.GetStream().Flush();

}

/// <summary>

/// Closes the transmission with the server.

/// </summary>

public void Close()

{

//close socket

c.Close();

}

}

}

Code Printout – ServerHanlder.cs – end

# רפלקציה

כשהתחלתי לעבוד על הפרויקט לא הייתי בטוח במה שאני הולך לעשות, כל פעם חששתי שאתקע בדרך ולא אוכל להתקדם, הפחד להתחיל משהו מבלי להיות בטוח איך בדיוק הוא הולך להיות גרם לי להמון חששות ותסכול. בהתחלה כן, בבניית השרת והלקוח נתקעתי בהמון בעיות שלא ידעתי איך לפתור כי לא למדתי את זה בשום מקום אחר לפני.. אך בעזרת המון רצון הצלחתי לעבור את החלק ההוא, כל בעיה בתחילת הפרויקט לקחה לי המון זמן של עבודה בכדי להבין איך בכלל ניגשים לבעיות בסדר גודל של פרויקט, ואיך אני בונה פרויקט כך שאוכל לכתוב עוד הרבה מבלי להרוס את מה שכבר כתבתי, אך ככל שהזמן התקדם וכך גם הפרויקט, למדתי המון על עבודה בפרויקטים, למדתי גם לדעתי את הדבר הכי חשוב שיש בניית פרויקטים, שזה פתירת תקלות בעזרת די-באג. בתחילת השנה לא ידעתי כלום בנוגע לתקשורת בין שתי תוכנות וכלום בנוגע לדי-באגינג.. כשביקשתי עזרה מחברים הם עזרו לי אך כל פעם אמרו לי שכשהפרויקט שלי יהיה מורכב יותר הם לא יוכלו לפתור לי הכל, מה שאישר את החשש שלי ולכן החלטתי לעשות את כולו בעצמי ולפתור את התקלות לבד, כשהבנתי איך עושים את זה, הפרויקט התחיל לרוץ לי בקצב טוב.. אפילו טוב יותר משתכננתי ולכן החלטתי שאני יעלה עוד קצת את הקצב ויבנה את כל מאגר המידע שלי בעצמי, חשבתי רבות על איך לעשות את זה בצורה כזאת שתיהיה דומה לאס-קיו-אל לייט וחשבתי על הדרך שאני משתמש בה כיום, בניית פורמט מסויים ואכסון של כל המידע בקבצי טקסט.

כשסיימתי לכתוב את המשחק וכשהתחלתי לראות כבר את סופו הרגשתי את ההרגשה של סיום הפרויקט בכיתה י' באסמבלי, כשאתה כותב פרויקט ומסיים אותו ושאתה יודע שעשית את כל זה בעצמך אתה מעריך דברים כאלה יותר, וגם את עצמך בניית הפרויקט לא הייתה דבר יותר מידי קשה אלא קצת מאתגר ומהנה, אומנם כן, אני לא מאלה שהיו כותבים פרויקט בסדר גודל כזה אם לא הייתי חייב אך כן בדיעבד אני לא מתחרט על כלום.

לסיכום, העבודה על הפרויקט הייתה לא רעה, אני מרגיש שהספקתי לעשות המון ואני גאה בפרויקט שאני מציג, אין לי שום ספקות בפרויקט ואני חושב שאני מגיש פרויקט מצוין... המסקנות שאני לוקח מהפרויקט הזה שלפעמים הצורה הכי טובה ללמוד משהו זה פשוט לעשות אותו עד שאתה מצליח, מבלי יותר מידי לחשוב איך ומה יהיה, אך כן לתכנן קדימה גם אם אתה לא בטוח שתספיק, ובכל מקרה להשאיר "מרווח אוויר" קטן שתוכל לתמרן בזמנים על מנת לא להיחנק מעבודה על הפרויקט.

# ביבליוגרפיה

## אתרים שהשתמשתי בהם

1. Stack Overflow - <https://stackoverflow.com/>
2. GitHub - <https://github.com/>
3. C# Corner - <https://www.c-sharpcorner.com/>
4. Code Project - <https://www.codeproject.com/>
5. Microsoft MSDN - <https://www.microsoft.com/he-il/>