|  |
| --- |
| Accounts(username,password,loggedin) |
| Username – שם המשתמש |
| Password - הסיסמה |
| Loggedin – האם המשתמש מחובר |

תיאור בסיס הנתונים:

טכנולוגיות בהם השתמשנו בפרוייקט

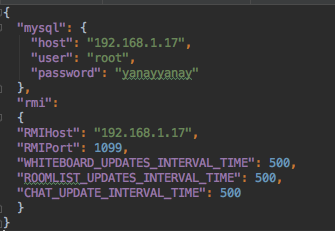
|  |  |
| --- | --- |
| שם הטכנולוגייה בה השתמשנו | פירוט |
| RMI – Remote Method Invocation | טכנולוגייה המאפשרת העברת מופע של אובייקט מרחוק, המחלקה צריכה לממש ממשק וככה אפשר להעביר את המופע בין מערכות שונות ולקרוא למתודות על אותו מופע בין מערכות שונות. |
| JDBC – Java Database Connectivity | הJDBC הוא API אשר מגדיר איך קליינט צריך לגשת למסד נתונים. במקרה שלנו השתמשנו בJDBC המובנה של Java והשתמשנו במסד הנתונים Mysql Server. |
| JavaFX | ספריית הגרפיקה שבה השתמשנו בפרוייקט. הספרייה מוכרת בשפת התכנות Java ויחסית פשוטה לשימוש. השתשמנו בFXML לכל חלון עיצוב, המאפשר בנייה של חלון העיצוב בעזרת תוכנה הנקראת Java Scene Builder. |
| CSS (Cascading Style Sheets) | פורמט המאפשר עיצוב ומילוי לאובייקטים הנוצרו על ידי הFXML ובעזרת הJava Scene Builder. |
| JSON | אובייקט נוח לפירסור נתונים, משמש בפרוייקט לטעינת הקונפיגורציה. |

תיאור הפרוייקט

**מחלקת Login**

זוהי המחלקה הראשית של הClient, על מנת להפעיל את הקליינט, מריצים את הMain של Login.java, המחלקה יורשת מApplication של JavaFX ולמעשה טוענת את LoginDesign.fxml שזה קובץ שעוצב בJava Scene Builder ואחראי על מסך העיצוב של ההתחברות. בנוסף, הFXML מתווך בין העיצוב לקוד, כלומר, כאשר בתוך הFXML מוגדרת מחלקת Controller בשם LoginController.java. בתוכה המתודות הבאות:

**OnConfigurationClick()** – כאשר לוחצים על הגלגל של ההגדרות למעשה נפתחת לנו אפשרות לבחור קובץ קונפיגורציה. הקובץ קונפיגורציה נראה כך:



הקובץ קונפיגורציה הוא למעשה קובץ json המכיל 2 מפתחות של mysql ושל rmi.

Host – כתובת האייפי של שרת הMysql

User – שם המשתמש של שרת הmysql

Password – הסיסמא של שרת הmysql

RMIHost – כתובת האייפי של שרת הRMI

RMIPort – הפורט של שרת הRMI

השאר יתוארו בהמשך. לאחר שטענו את הקובץ הפונקציה קוראת לשיטה updateConfigurationFromFile שנמצאת במחלקה DBFinals, שאחראית על טעינת הנתונים בקובץ ושימוש בהם על מנת להתחבר לשרתים הרלוונטים. במידה והחיבור לא מצליח מוצגת הודעה למשתמש

**מחלקת LoginController**

**onRegisterClick** – הפונקציה נקראת כאשר לוחצים על כפתור ההרשמה. במידה ואחד מהנתונים חסר מוצגת הודעה למשתמש. במידה ומנסים להתחבר למשתמש שכבר רשום מוצג גם במקרה הזה הודעה למשתמש.

לאחר שההרשמה הצליחה מוצגת הודעה למשתמש שהמשתמש נוצר בהצלחה.

**onLoginMouse** – הפונקציה נקראת כאשר לוחצים על כפתור ההתחברות. הפונקציה מאמתת עם הsql בעזרת rmi את הפרטים. במידה והמשתמש כבר מחובר או שהסיסמא לא נכונה או שהמשתמש לא קיים, מוצגת הודעה למשתמש, אחרת החיבור הצליח ועוברים למסך הבא.

**מחלקת Lobby**

Lobby היא מחלקה שיורשת מRunnable ורצה כThread נפרד. התפקיד שלה זה לשלוף בכל כמות זמן (המוגדרת בקבוע ROOMLIST\_UPDATES\_INTERVAL\_TIME) את רשימת החדרים מהRMI ובמידה ונוסף חדר חדש לעדכן גם אצל הClient. היא משווה בין רשימת החדרים של הקליינט לבין רשימת החדרים העדכנית בRMI והיא אחראית על עדכון העיצוב.

**מחלקת State**

המחלקה State היא מחלקה המכילה רק משתנים סטטיים. מכיוון שאנו עוברים כל הזמן בין חלונות וכל מחלקה מייצגת חלון אחר, צריך איזשהו מחלקה ששומרת בזמן ריצה את המצבים המשתנים וככה היא יכולה לשמור מידע גם אם עברנו בין חלונות. למשל, כאשר אנחנו עושים Login עם שם משתמש וסיסמא, אנחנו רוצים לשמור את המשתמש שעשה Login גם כאשר עברנו לחלון של הRoom בשביל להציג אותו ברשימת המחוברים לחדר. לכן המחלקה State עוזרת לטיפול בעניין זה. במחלקה State מוגדרים המשתנים הסטטיים הבאים:

**chatUpdateTimer** – מהווה את הטיימר שרץ כל חצי שנייה (אפשר לשנות בקונפיגורציה) והתפקיד שלו זה להשוות את הChat של הקליינט לChat של הסרבר בעזרת הRMI ובמידה וצריך לעדכן את הצאט של הקליינט הוא דואג לעשות זאת. הוא רץ בתהליכון נפרד.

**graphicsUpdateTimer** – אותו דבר כמו הצ'אט רק בשביל הגרפיקה. הוא בודק אם התבצע עדכון לגרפיקה (למשל כאשר משתמש אחר מצייר משהו) ובמידה וכן הוא שולף את העדכון מהRMI ומעדכן אצל הקליינט.

**mainController** – הפנייה לController שאחראי על המסך שכרגע מוצג למשתמש, בעת מעבר בין חדרים הController מתעדכן.

**roomManager** – זוהי בעצם ההפניה למופע של אובייקט ממחלקת RoomManager, הפנייה זו היא הפנייה שרשומה בRMI וכך למעשה אפשר לקרוא למתודה מסויימת שמשנה ערכים במופע, ונוכל לראות את זה בין כל הקליינטים. על הroomManager מבוסס כל רעיון הRMI והעברת ההודעות והגרפיקה בין משתמשים. הטעינה שלו נעשית כאשר אנו טוענים את קובץ הקונפיגורציה (הוא בודק מאיפה לעשות lookup ע"י קריאת הערכים המתאימים בקובץ הקונפיגורציה).

**JdbcManager** – גם הוא נטען בקונפיגורציה ואחראי על המתודות של הSQL, קליינט יכול להשתמש בו על מנת לקרוא למתודה אשר בודקת אם שם המשתמש שהזין נכון, וכו.

**lastClick** – שומר את הנקודה האחרונה עליה המשתמש לחץ. הכרחי למשל בציור קו כי המשתמש צריך לבחור 2 נקודות.

**lastTimeUpdatedGraphics** – שומר את התאריך האחרון בו התבצע עדכון במשתמש הספציפי לגרפיקה. נועד על מנת להשוות בינו לבין התאריך האחרון של העדכון שרשום בRMI

**lastTimeUpdatedChat** – כמוסבר למעלה.

**lastTimeUpdatedRoomList** – כמוסבר למעלה.

**drawState** – מייצג את הכפתור עליו לחצנו בToolBox ואת מצבו.

**currentToolBoxItemClicked** – מייצג באופן גרפי את הכפתור עליו לחצנו על מנת לסמן את צבעו בכחול.

**Username** – שם המשתמש של הקליינט.

**roomName** – שמו של החדר אליו המשתמש מחובר.

**מחלקת LobbyController**

מחלקת LobbyController זוהי המחלקה המקשרת בין העיצוב של הLobby (שילוב של הFXML והCSS) לבין הקוד עצמו. במחלקה בעצם מימוש של כל כפתורי הלובי ובניהם הפונקציות הבאות:

**joinRoom** – הצטרפות לחדר שסומן תוך כדי עדכון של המצבים הרלוונטים בState.

**refreshList** – רענון רשימת החדרים. מכיוון שהחדרים מתעדכנים באופן אוטומטי כל כמות זמן מסויימת זה לא כל כך רלוונטי, אולם אם מגדילים את זמן האינטרוול זה כן יכול להיות רלוונטי.

**addToRoomList** – מוסיפה חדר לרשימת החדרים (מבחינת הUI).

**resetRoomList** – מאפסת את רשימת החדרים.

**createRoom** – לאחר לחיצת על כפתור היצירת חדר חדש, מתבצעת קריאה לפונקציה זו שפותחת חלון שבו כותבים את שם החדר החדש. במידה והחדר לא קיים הפונקציה פותחת את החדר החדש ומוסיפה אותו לרשימת החדרים, ובנוסף מוסיפה את החדר לRMI.

**disconnect** – במידה והמשתמש לוחץ על כפתור ההתנתקות מתבצעת קריאה לפונקציה זאת שמעדכנת במסד נתונים שהמשתמש התנתק ומעבירה את המשתמש לחלון הlogin.

**logout** – מעדכנת במסד נתונים שהמשתמש התנתק.

**מחלקת Room**

מחלקה שהיא Serializable שעוברת בRMI ותפקידה לייצג חדר.

משתני המחלקה:

name – שם החדר

**roomAdminUsername** – שם המשתמש של מי שיצר את החדר

chat – אובייקט Serializable מסוג chat שאחראי על ניהול הצ'אט בחדר

**ArrayList<Shape>shape** – מערך של כל הצורות שיש בחדר על לוח הwhite board

**undoShapes** – מערך של כל הצורות שיש בחדר ועשינו להם undo על מנת לשמור אותם למקרה של redo.

**ArrayList<String>clients** – רשימת כל הקליינטים המחוברים לחדר (על מנת להציג אותם לכל משתמש)

**graphicsUpdateTime** – הזמן האחרון שבו עשינו עדכון לגרפיקה

**roomListUpdateTime** – הזמן האחרון שבו עשינו עדכון לרשימת המחוברים לחדר

בנוסף לכל תכונות המחלקה יש פונקציות שמספקת המחלקה, אך ניתן להסיק מה הפונקציות עושות על ידי השם שלהם.

**מחלקת RoomController**

מחלקת RoomController זוהי המחלקה המקשרת בין העיצוב של החדר (שילוב של הFXML והCSS) לבין הקוד עצמו. במחלקה בעצם מימוש של כל כפתורי החדר (כולל הToolBox) וגם את הפונקציות הבאות:

**clearWhiteBoard** – מנקה את הלוח

**drawLine** – מצייר קו בהינתן 2 נקודות וצבע

**drawRectangle** – מצייר מרובע בהינתן נקודה (שהיא הנקודה הכי שמאלית עליונה במרובע) אורך המרובע, רוחב המרובע.

**drawCircle** – מצייר עיגול בהינתן נקודה שמאלית עליונה של העיגול, רדיוס אורך העיגול, רדיוס רוחב העיגול וצבע.

**drawContinuousLine** – מציירת קו רציף בהינתן מערך של כל הנקודות המייצגות אותו, וצבע.

**drawText** – מציירת טקסט בהינתן טקסט צבע ונקודה איפה לצייר את הטקסט.

**appendChat** – מוסיפה לצ'אט הכללי (בRMI) את ההודעה שהתקבלה כפרמטר לפונקציה.

**initDraw** – צובעת את מסגרת הCanvas בשחור.

on(Shape)Click – מסמנת את מסגרת הכפתור בtoolbox בכחול.

**logoutClicked** – מעבירה את המשתמש למסך בחירת החדרים ומנתקת אותו מהחדר תוך כדי פרסום הודעה לכלל המשתמשים בחדר שהמשתמש עזב.

**enterClick** ברגע שהמשתמש לחץ על enter כשהוא כותב הודעה בצ'אט, הפונקציה מפרסמת את ההודעה שכתב בעזרת הRMI.

**onUndoClicked** – עושה undo.

**onRedoClicked** – עושה redo.

**מחלקת ServerController**

מחלקת ServerController זוהי המחלקה המקשרת בין העיצוב של הסרבר (שילוב של הFXML והCSS) לבין הקוד עצמו. במחלקה בעצם מימוש של טעינת פרטי השרתים בקונפיגורציה (שרת הSQL ושרת הRMI), ויצירה של שרת הRMI.

**מחלקת הJDBCManager**

מחלקה המועברת בRMI ומטרה לספק שאילתות (שהם למעשה הפונקציות):

**createUser** – יצירת משתמש חדש

**checkLogin** – בדיקה האם משתמש מחובר.

**logInOutUser** – עדכון בשאילתא שהמשתמש התנתק או התחבר.