Cient Server Shticell

בונוס שמומש:

2. ניתוח דינמי מרובה משתנים. (יש Slider אחד אך באמצעות לחיצה על הכפתור dynamic analysis ניתן להחליף את התא שכרגע מושפע מהSlider וכך לבצע את הניתוח על כמה תאים שרוצים)

סקירה כללית

הפרויקט הוא מערכת לניהול גיליונות אלקטרוניים אשר יכולה להכיל מספר משתמשים וגיליונות אלקטרוניים ומאפשרת לשתף ולערוך גיליונות אלקטרוניים ע" משתמשים שונים במקביל, כאשר הגיליונות נשמרים בשרת והמשתמשים השונים הם קליינטים של אותו שרת.

מבנה הפרויקט:

הפרוייקט מחולק לחמישה מודולים שונים אשר התלויות ביניהם מוצגות בסכמה הבאה כאשר חץ מסמל תלות(dependency):

Client → DTO

Client → serializer

webServer → DTO

webServer → serializer

webServer → Engine

Engine → DTO

serializer → DTO

serializer → engine (for concrete classes during deserialization)

כאשר מודול הwebServer והengine מהווים את ה**שרת** מודול הclient מהווה את ה**לקוח** והמודולים DTO ו-serializer **מקשרים בין השרת ללקוח.**

המודולים השונים במערכת:

- **engine**
  - מכיל את ליבת הלוגיקה והמימוש של הממשקים הנמצאים בתוך מודול הDTO
- **client**
  - כולל את רכיבי ממשק המשתמש המתקשרים עם השרת בעזרת בקשות HTTP.
- **DTO**
  - כולל את הממשקים ומידע המועבר בין השרת ללקוח בצורת json.
- **serializer**
  - כולל את לוגיקת הserialization והdeserialization בין הלקוח לשרת מכיל serializer ו-deserializer לאובייקטים השונים המועברים ביניהם כמו כן, מכיל את המופע של GSON(ספריית צד שלישי לserialization) בו משתמשים בכל המערכת על מנת להעביר מידע בין השרת ללקוח.
- **webserver**
  - מודול web המכיל את servlets השונים איתם הלקוח מתקשר ושבעצם מנגישים את יכולות מנוע המערכת למשתמש.

מחלקות עיקריות ותפקידיהן

מודול Dto

מחלקות עיקריות:

- **Logic INTERFACE**
  - **תיאור** : הממשק הזה נועד ליהיות אובייקט הDTO הראשי המייצג גיליון אלקטרוני מסוים. לכל גיליון יש מופע משלו של הממשק הזה והוא ממומש במסגרת מנוע המערכת. ממשק המשתמש מתנהל עם גיליון מסויים אך ורק דרך ממשק זה ולא מכיר את המימוש הספציפי שלו. ממשק זה מכיל מטודות כגון: החזרת האורך והרוחב של גיליון וגודלו וגם החזרת גיליון נוכחי או גיליון לפי גרסה, כמו כן מכיל גם את המטודות אשר מפעילות את המנוע לבצע פקודות שהמשתמש ביקש כמו לעדכן תא או לצפות בתא.
  - **שיטות עיקריות**:

- `int getRows()`:
  מחזירה את מספר השורות בגיליון.

- **int getColumns()**:
  מחזירה את מספר העמודות בגיליון.
- **int getWidth()**:
  מחזירה את רוחב הגיליון.
- **int getHeight()**:
  מחזירה את גובה הגיליון.
- **String getSheetName()**:
  מחזירה את שם הגיליון.
- **ReadonlySheet getCurrentSheet()**:
  מחזירה את הגיליון הנוכחי.
- **ReadonlySheet getSheetByVersion(int version)**:
  מחזירה גיליון לפי גרסה.
- **Logic loadSheet(String filePath**:
  טוענת גיליון מקובץ.
- **Logic loadSheetFromXmlContent(InputStream inputStream, String uploaderName)**:
  טוענת גיליון מקובץ כאשר נתוני הקובץ מועברים כ-stream של בתים.
- **void displaySheet(int version)**:
  מציגה גיליון לפי גרסה.
- **ReadonlyCell getCellForDisplay(int version, String rowAndColumn)**:
  מחזירה תא להצגה לפי גרסה ומיקום בפורמט אקסל
- **void updateCell(int version, String rowAndColumn, String value)**:
  מעדכנת את ערך התא לפי גרסה וקואורדינטות.
- **List<ReadonlySheet> getDifferentSheetVersionsForDisplay()**:
  מחזירה רשימה של גרסאות גיליון שונות להצגה.
- **Void createNewRange(String rangeName, int rowStart, int columnStart, int rowEnd, int columnEnd)**:
  יוצרת טווח חדש, זורקת חריגות אם הטווח כבר קיים, אם הקואורדינטות מחוץ לטווח או אם אינדקס ההתחלה נמוך מאינדקס הסיום.
- **void removeRange(String rangeName)**:
  מסיר טווח לפי שם הטווח. זורק חריגה אם הטווח בשימוש.
- **ReadonlySheet FilterSheet(List<String> selectedUniqueValues, int columnNumber, Coordinate topLeft, Coordinate bottomRight, ReadonlySheet sheetToFilter)**:
  מסננת את הגיליון לפי רשימת ערכים ייחודיים שנבחרו, מספר עמודה, וקואורדינטות בפינה העליונה והשמאלית ובפינה התחתונה והימנית של הגיליון לסינון. זורק חריגה אם הקואורדינטות מחוץ לטווח- מחזיר גיליון חדש מסונן.
- **sortSheet(List<String> selectedColumns, Coordinate topLeft, Coordinate bottomRight, ReadonlySheet sheetToSort)**:
  ממיינת את הגיליון לפי רשימת עמודות שנבחרו, וקואורדינטות בפינה העליונה והשמאלית ובפינה התחתונה והימנית של הגיליון למיון. זורקת חריגה אם הקואורדינטות מחוץ לטווח מחזירה גיליון חדש ממויין.

- `Map<String, Permission> getPermissions()`:
  מחזיר מפת מחרוזות להרשאות, כאשר המחרוזת היא שם המשתמש וההרשאה היא סוג ההרשאה של המשתמש עבור גיליון זה.
- `List<PermissionRequest> getPermissionRequestsForSheets()`:
  מחזיר רשימה של בקשות הרשאה עבור הגיליון הזה, כאשר כל בקשה מכילה מידע על המשתמש שביקש הרשאה ועל סוג ההרשאה המבוקשת.
- `String getUserPermission(String userName)`:
  מחזיר את סוג ההרשאה של משתמש מסוים עבור גיליון זה לפי שם המשתמש שניתן כפרמטר.
- `String getOwnerName()`:
  מחזירה את שם הבעלים של הגיליון.

- **ReadonlySheet INTERFACE**
  - **תיאור**: ממשק המכיל פעולות לקריאה ושליפת פרטים(readonly) של גיליון ספציפי, שכבת הUI מכירה גיליון רק דרך הממשק הזה. הממשק מכיל מטודות כמו שליפה של התאים הפעילים בגיליון(תאים לקריאה בלבד), גרסת הגיליון, מספר התאים שהשתנו בגיליון זה בהשוואה לגרסה הקודמת של הגיליון וכדומה.
  - **שיטות עיקריות**:

- **int getVersion()**:
מחזירה את גרסת הגיליון.

- **ReadonlyCell getCell(int row, int column)**:
מחזירה תא לפי השורה והעמודה שלו.

- **List<ReadonlyCell> getActiveCells()**:
מחזירה רשימה של תאים פעילים בגיליון.

- **int getChangedCellsCount()**:
מחזירה את מספר התאים שהשתנו בגרסה הנוכחית.

• **addNewRange(String range, int rowStart, int columnStart, int rowEnd, int columnEnd):**
מ מוסיפה טווח חדש לגיליון הנוכחי זורקת חריגות אם הטווח כבר קיים, אם הקואורדינטות מחוץ לטווח, או אם אינדקס ההתחלה נמוך מאינדקס הסיום.

• **removeRange(String range):**
מסירה טווח לפי שמו מהגיליון הנוכחי זורקת חריגה אם הטווח בשימוש כרגע.

• **getCellsThatAreUsingThisRange(String range):**
מחזירה סט של תאים בגיליון הנוכחי שמשתמשים בטווח שצוין.

- **String getUpdater():**
מחזירה את המעדכן הנוכחי של הגיליון.

- **List<String> getNumericCells():**
מחזירה רשימה של תאים מספריים של הגיליון

- **Sheet INTERFACE**
  - **תיאור:**
    ממשק היורש(extends) מהממשק ReadOnlySheet ומתאר גיליון בודד ומאפשר לבצע פעולות עדכון על הגיליון הזה המנוע מממש את ממשק הזה ובכך יכול לעדכן תאים ולבצע את המח"מ בגיליון זה.
  - **שיטות עיקריות:**

- **updateCellValueAndCalculate(int row, int column, String value):**
מבצעת עדכון בגיליון הנוכחי של תא בשורה והעמודה שהתקבלה ולפי הערך שהתקבל. ולאחר מכן מתחילה את המח"מ של הגיליון שבמהלכו מתחשבים כל התאים בגיליון מחדש על מנת לשקף את התלויות החדשות בין התאים בגיליון במידה ונוצרו. כמו כן, מתריעה גם אם נוצרה תלות מעגלית בין התאים בגיליון.

- **void incrementVersion():**
מעלה את הגרסה של הגיליון.

- **Logic getLogic():**
מחזיר את הלוגיקה(אובייקט המשקף את מצב כל הגיליונות במערכת כולל את של גיליון זה) הקשורה לגיליון.

- **void setVersion(int version):**
מעדכן את הגרסה של הגיליון לערך ספציפי .

- **void setLogic(Logic logic):**
מעדכן עבור גיליון זה את הלוגיקה (אובייקט המאכסן בתוכו את כל הגרסאות של אותו גיליון ופרטים נוספים אודות הגיליון).

- **void setUpdater(String updater):**
מעדכן את המעדכן הנוכחי של הגיליון לשם שניתן כפרמטר.

- **ReadOnlyCell INTERFACE**
  - **תיאור:**
    ממשק אשר מימושו מאפשר גישת קריאה בלבד לפרטי תא מסוים בגיליון. מכיל מטודות כמו החזרת מיקום התא בגיליון(קוארדינטה) רשימת התאים שהוא תלוי/משפיע עליהם וכדומה.
  - **שיטות עיקריות:**

- **Coordinate getCoordinate()**
מחזירה את הקואורדינטות של התא

- **String getOriginalValue()**
מחזירה את הערך המקורי של התא.

- **EffectiveValue getEffectiveValue()**
מחזירה את הערך האפקטיבי של התא.

- **boolean calculateEffectiveValue()**
מתחילה חישוב מחדש של ערכו האפקטיבי של התא. מחזירה אמת אם הערך השתנה מהערך הקודם ושקר אחרת.

- **int getVersion()**
מחזירה את גרסת התא.

- **Set<ReadonlyCell> getDependsOn()**
מחזירה את קבוצת התאים שהתא תלוי בהם.

- **Set<ReadonlyCell> getInfluencingOn()**
מחזירה את קבוצת התאים שתלויים בתא זה

- **String getLastUpdater():**
מחזירה את שם המעדכן האחרון של תא זה

- **Cell INTERFACE**
  - **תיאור:**
    ממשק היורש(extends) מ-ReadOnlyCell ומוסיף פעולות עריכה של תא כמו לעדכן את הערך המקורי של התא ולעלות גרסה שלו.
  - **שיטות עיקריות:**

- **void setCellOriginalValue(String value)**
  קובע את הערך המקורי של התא.
- **void incrementVersion()**
  מעלה את הגרסה של התא.
- **void setVersion(int version)**
  קובע את הגרסה של התא.
- **String getDependsOnRange():**
  מחזירה את שם הטווח שתא זה נסמך עליו. אם אין טווח כזה, מחזירה null.
- **Set<String> getInfluencingOnRanges():**
  מחזירה סט של שמות טווחים שתא זה נמצא בתוכם.

- **void setSheet (ReadonlySheet sheet):**
  מעדכן עבור תא זה את הגיליון בו הוא נמצא.

- **void setUpdater(String updater):**
  מעדכן את המעדכן הנוכחי של תא זה.

- **EffectiveValue INTERFACE**
  - **תיאור:**
    ממשק המתאר את הערך האפקטיבי של התא ומכיל מטודות המאפשרות להחזיר את סוג התא של התא הנוכחי(מבין כלל סוגי התאים הקיימים במערכת) ואת ערכו כמו כן, מכיל מטודה המאפשרת לבדוק האם הטיפוס שקיבלנו בזמן ריצה מתאים לטיפוס המצופה, המטודה מאפשרת עבודה עם סוגי טיפוסים שונים, כל תא בגיליון מכיל אובייקט שמממש מחלקה זאת.
  - **שיטות עיקריות:**

- • **CellType getCellType()**
  מחזיר את סוג התא.
- • **Object getValue()**
  מחזיר את הערך של התא.
- • **<T> T extractValueWithExpectation(Class<T> type)**
  מטודה המאפשרת לבדוק האם הטיפוס שקיבלנו הוא הטיפוס שציפינו או יכול ליהיות מומר אליו, אם כן נחזיר מופע של הטיפוס המתאים אחרת נזרוק שגיאה.

- **Coordinate Class**
  - **תיאור:**
    מחלקה המתארת מיקום של תא בגיליון מאפשרת לשלוף את השורה והעמודה הרלוונטיים ומאפשרת גם להשוות בין שתי מיקומים בעזרת מטודות equals וhashcode שמומשו בה.
  - **שיטות עיקריות:**

- **public int getRow()**
מחזירה את אינדקס השורה של הקואורדינטה.

- **public int getColumn()**
מחזירה את אינדקס העמודה של הקואורדינטה.

- **CoordinateFactory Class**
  - **תיאור:**
    מחלקה האחראית על יצירת אובייקטים של מחלקת Coordinate הנ"ל. מכילה מאין cache של קוארדינטות שכבר נוצרו ובכך מונעת את יצירתם מחדש. בנוסף שימוש

במחלקה ייעודית ליצירת הקוארדינטות מאפשר הוספת לוגיקה נוספת לתהליך יצירתם בעתיד ביתר קלות.

- **שיטות עיקריות**:

- **public static Coordinate createCoordinate(int row, int column)**
  יוצר ומחזיר אובייקט קואורדינטה חדש עם השורה והעמודה שצוינו

- **Expression INTERFACE**

  - **תיאור**:
    ממשק המתאר ביטוי מסוג כלשהו במערכת הממשק מאפשר להתייחס לביטויים שונים במערכת בצורה אחידה וכל סוג ביטוי מממש אותו בצורה המתאימה לו, סוג ביטוי לדוגמא הוא ביטוי הMINUS שמכיל פונקציה ושתי ארגומנטים ביניהם הוא מבצע חיסור. הממשק מכיל מטודה לשליפת סוג התא המתאים לביטוי ולחישוב ערכו הסופי של הביטוי(בדוגמא הנ"ל את תוצאת החיסור)

  - **שיטות עיקריות**:

- • **EffectiveValue eval(ReadonlySheet sheet)**
  מחשב את הביטוי בהתבסס על גיליון לקריאה בלבד ומחזיר את הערך האפקטיבי של הביטוי
- • **CellType getFunctionResultType()**
  מחזיר את סוג התא המתאים לתוצאת הביטוי.

- **PermissionRequest INTERFACE**

  - **תיאור**:
    ממשק המגדיר בקשת הרשאה עבור גיליון מסויים הממשק מכיל פעולות המאפשרות לקבל את נתוני הבקשה.

  - **שיטות עיקריות**:

- **Permission getPermission():**
מחזירה את ההרשאה המבוקשת עבור הגיליון.

- **String getSubmitterName():**
מחזירה את שם המגיש של הבקשה.

- **String getSheetName():**
מחזירה את שם הגיליון.

- **String getSheetOwnerName():**
מחזירה את שם הבעלים של הגיליון.

- **PermissionRequestStatus getStatus():**
מחזירה את מצב בקשת ההרשאה.

- **Permission ENUM**
  - **תיאור:**
    ENUM המתאר את סוגי ההרשאות הקיימות עבור גיליונות במערכת.
  - **ערכי הPermission במערכת:**

- ***OWNER***
- ***READER***
- ***WRITER***
- ***NONE***

מודול Client

מחלקות עיקריות:

- **MainController CLASS**
  - **תיאור:**
    מחלקה המתארת את הבקר(controller) המרכזי של מסך הצגת גיליון. המחלקה מאגדת בתוכה תתי קומפוננטות נוספות של הUI ומכילה גם מופע של Dto עבור הגיליון המוצג כרגע(ממשק Logic) ואחראית לתקשורת בינו לבין רכיבי הUI האחרים.
  - **שיטות עיקריות:**

- **`void initialize():`**
מאתחלת את הרכיבים העיקריים של ה-GUI, כולל טעינת קבצי FXML שונים והגדרת בקרים עבור חלקים שונים ביישום.

- **`void handleCellClick(Label cellLabel, int row, int col):`**
מטפלת בלחיצה על תא בגיליון מסמנת אותו ואת התאים המסתמכים עליו שהוא מסתמך עליהם

- **`ObservableList<Integer> getAllSheetsVersions():`**
מחזירה את כל גרסאות הגיליונות הזמינים להצגה.

- **`void applyColumnChanges(int column, String alignment, double width):`**
מיישמת שינויים בעמודה ספציפית, כמו יישור ורוחב.

- **`void applyRowChanges(int row, double height):`**
מיישמת שינויים בשורה ספציפית, כמו גובה.

- **void sendRequestToUpdateCellValue(String rowAndColumn, String value):**

יוצרת ושולחת בקשת HTTP לשרת על מנת לעדכן תא בגיליון המוצג כרגע.

- **void openVersionDisplayer(int version):**

מציגה גרסה נוספת של הגליון בחלון הנוכחי.

- **void sendRequestToCreateRange(String rangeName, String topLeftCell, String bottomRight):**

יוצרת ושולחת בקשת HTTP לשרת על מנת ליצור טווח תאים חדש בגיליון המוצג כרגע למשתמש.

- **void rangeSelectionChanged(String selectedRange):**

מטפלת באירוע כאשר טווח נבחר, מסמנת את הטווח ב-UI.

- **void sendRequestToRemoveRange(String selectedRange):**

יוצרת ושולחת בקשת HTTP לשרת על מנת למחוק טווח תאים מסויים מהגיליון המוצג כרגע למשתמש.

- **void sendRequestTofilterSheet(List<String> selectedUniqueValues, String SelectedColumn, Coordinate topLeft, Coordinate bottomRight):**

יוצרת ושולחת בקשת HTTP לשרת על מנת לסנן את הגיליון בהתאם לערכים הנבחרו על ידי המשתמש, אם הסינון עבר בהצלחה השרת מחזיר את הגיליון המסונן.

- **void displayFilteredOrSortedSheet(ReadonlySheet filteredSheet, boolean isFiltered):**

מציגה גיליון מסונן או ממוין בחלון חדש.

- **void sendRequestsortSheetByColumns(List<String> selectedColumns, Coordinate topLeft, Coordinate bottomRight):**

יוצרת ושולחת בקשת HTTP לשרת על מנת למיין את הגיליון בהתאם לערכים הנבחרו על ידי המשתמש, אם הסינון עבר בהצלחה השרת מחזיר את הגיליון הממויין.

- **void openDynamicAnalysis():**

המטודה פותחת חלון אשר מאפשר למשתמש לבחור תא אשר עליו יופעל הניתוח הדינאמי (כלומר הslider יפעל בהקשרו וגם את הפרמטרים לאותו slider) בסיום הבחירה בהנחה ותקינה יפתח חלון חדש בו יוצג הגיליון הנוכחי עם slider המאפשר לעשות ניתוח דינאמי על התא שנבחר.

- **void updateRanges()**

מעדכנת את רשימת הטווחים המוצגים למשתמש.

- **void displaySheetAccordingToPermissionsAndVersion()**

מציגה את הגיליון בהתאם להרשאות של אותו משתמש בין אם הם הרשאות כתיבה או עריכה(חוסמת או פותחת רכיבי ממשק משתמש בהתאם להרשאות)

- **`void updateDisplayedSheet()`**

מטודה האחראית לעדכון הגיליון המוצג למשתמש וגם לעדכון חיווים כמו העובדה שקיימת גרסה עדכנית יותר של הגיליון מזאת המוצגת כרגע. בנוסף אחראית גם לאפשר פעולות מסויימות בהתאם להרשאות המשתמש הנוכחי.

- **`void switchToDashboardView()`**

מטודה האחראית למעבר מתצוגת גיליון ספציפי למסך הdashboard המאפשר צפייה בכלל הגיליונות במערכת.

- **DashBoard CLASS**
  - **תיאור:**

מחלקה האחראית על תצוגת מסך הDashboard של המערכת, מסך המציג את כלל הגיליונות הקיימים במערכת וההרשאות של המשתמש הנוכחי עבורם בנוסף מכיל גם את כל בקשות ההרשאה עבור כל אחד מהגיליונות ונותן אופציה להציג גיליון ולבקש/הרשאה אליו. מחלקה זאת מפעילה את תהליך העדכון האוטומטי המתבצע כל שתי שניות בו הלקוח הספציפי מבקש מהשרת באופן אוטומטי לשלוח לו את המידע הקיים אצלו על כלל הגיליונות במערכת ומחלקה זאת מחליטה מה מהמידע המתקבל לעדכן אצל הלקוח.

  - **שיטות עיקריות:**

- `populatePermissionTypeComboBox()`:
ממלאת את תיבת הבחירה של סוגי ההרשאות.
- `populateAckDenyComboBox()`:
ממלאת את תיבת הבחירה של אישור/דחייה.
- `changeViewSheetButtonAccordingToPermission(String permission)`:
מפעילה או משביתה את כפתור הצגת הגיליון בהתאם להרשאה של המשתמש הנוכחי.
- `refreshPermissionRequestDataList(Logic selectedLogic, String uploadersName)`:
מרעננת את רשימת בקשות ההרשאה.
- `sendRequestForPermissionAckOrDeny(int selectedRequestNumber, ApprovalOption option, String sheetName, String uploadersName, String submittersName)`:
שולחת בקשה לשרת לאישור או דחיית בקשת הרשאה.
- `ackDenyButtonActionListener(ActionEvent event)`: מטפלת בפעולת כפתור האישור/דחייה
.
- `loadSheetButtonActionListener(ActionEvent event)`:
מטפלת בפעולת כפתור טעינת הגיליון.
- `viewSheetActionListener(ActionEvent event)`:
מטפלת בפעולת כפתור הצגת הגיליון.
- `submitRequestActionListener(ActionEvent event)`:
מטפלת בפעולת כפתור שליחת הבקשה.
- `switchToSheetDisplay(Logic selectedLogic)`:

מחליפה לתצוגת הגיליון.

- `sendRequestForPermissionChange(String OwnerName, String sheetName, Permission permission)`:
שולחת בקשה לשרת להוסיף בקשה הרשאות חדשה עבור גיליון.
- `sendRequestForFileUpload(File selectedFile)`:
שולחת בקשה לשרת לטעון נתוני גיליון חדש מקובץ.
- `updateUserLogicMapData(Map<String, Map<String,Logic>> newUsersListWithLogic)`:
מעדכנת את נתוני כלל הגיליונות במערכת אצל הלקוח(משווה בין הנתונים הקיימים אצל הלקוח לבין הנתונים החדשים שהתקבלו ומעדכנת רק נתונים החסרים אצל אותו לקוח)
- `updateSheetDataList(Logic newLogic, String uploadersName)`:
מעדכנת את הגיליונות המוצגים במערכת
- `insertNewSheets(Logic oldLogic, Logic newLogic)`:
מוסיפה גיליונות חדשים עבור הלקוח במידת הצורך
- `updateSheetPermissionForCurrentUser(Map<String, Permission> newPermissionsForSheets, Map<String, Permission> oldPermissionsForSheets, Logic newLogic, String username)`:
מעדכנת את ההרשאות של המשתמש הנוכחי לכלל הגיליונות במערכת
- `updatePermissionRequests(Logic oldLogic, Logic newLogic, String sheetName, String username)`:
מעדכנת את בקשות ההרשאה עבור גיליון
- `close()`:

סוגרת את הבקר, עוצרת את הטיימר ומכבה את לקוח הHTTP.

- **SheetDisplayerController CLASS**

  - **תיאור:**

    מחלקה האחראית המתארת את הבקר(controller) האחראי על הצגת הגיליון בממשק הגרפי של המערכת ואינטרקצית המשתמש עם התאים שבו. היא מטפלת באכלוס הגריד בנתוני תאים, ביצוע שינויים בשורות ובעמודות, וסימון תאים על פי קריטריונים שונים.

  - **שיטות עיקריות**:

- **`void setSheetData(ReadonlySheet displayedSheet, int rows, int columns, int columnWidth, int height, boolean keepStyle)`:**
מגדירה את הנתונים לגיליון שיוצג. מאכלסת את הגריד בנתוני הגיליון המסופקים ומיישמת את המידות והסגנונות שנבחרו.

- **`void setMainController(MainController mainController)`:**
מגדירה את הבקר הראשי עבור התצוגה של הגיליון בעצם יוצרת את הקשר לבקר הראשי.

- **void populateGrid(ReadonlySheet displayedSheet, int rows, int columns, int columnWidth, int height, boolean keepStyle):**
מאכלסת את הגריד בנתוני תאים מהגיליון המסופק. מטפלת ביצירה ובעיצוב של תוויות תאים וכותרות.

- **void handleCellClick(Label cellLabel, int row, int col):**
מטפלת באירוע כאשר תא נלחץ. ממנפת את הטיפול לבקר הראשי.

- **void internalApplyColumnChanges(int column, String alignment, double width):**
מיישמת בפועל שינויים ויזואליים בעמודה ספציפית, כגון יישור ורוחב.

- **void internalApplyRowChanges(int row, double height):**
מיישמת בפועל שינויים ויזואליים בשורה ספציפית, כגון גובה.

- **void markDependsAndInfluencingOnCells(Set<ReadonlyCell> dependsOn, Set<ReadonlyCell> influencingOn):**
מסמנת תאים שתלויים בתאים אחרים או משפיעים עליהם, ומעדכנת את הסגנון שלהם בהתאם.

- **void clearDependsAndInfluencingOnCells():**
מנקה את הסימונים של תאים שתלויים בתאים אחרים או משפיעים עליהם.

- **void markRange(Set<Coordinate> rangeCoordinates):**
מסמנת טווח של תאים בהתאם לקואורדינטות המסופקות.

- **void clearRangeMark():**
מנקה את הסימונים של טווח של תאים.

- **<u>UpperSectionController CLASS</u>**
  - **תיאור:**
    מחלקה המתארת את הבקר המנהל את החלק העליון של הGUI ומטפלת באינטרקציות של המשתמש עם אזור זה של הGUI כגון צפייה בגרסאות שונות של הגיליון, עדכון וצפייה בערכי תאים ושינויי skins של המערכת.
  - **שיטות עיקריות:**

- **void applyDarkModeSkin(ActionEvent event):**
מיישמת את עיצוב מצב הלילה על היישום על ידי עדכון הגדרות הסגנון.

- **void applyModernBlueSkin(ActionEvent event):**
מיישמת את העיצוב הכחול המודרני על היישום על ידי עדכון הגדרות הסגנון.

- **void applyDefaultSkin(ActionEvent event):**
מיישמת את העיצוב הדיפולטי על היישום על ידי עדכון הגדרות הסגנון.

- **`void setMainController(MainController mainController)`**:
מגדירה את הבקר הראשי עבור החלק העליון, מאפשרת לו אינטראקציה עם הלוגיקה העיקרית של היישום.

- **`void handleSheetVersionChange(ActionEvent event)`**:
מטפלת באירוע כאשר המשתמש מחפש להציג גרסה ישנה.

- **`void openSheetDisplayWindow(int version)`**:
מציגה את גרסת הגיליון המצוינת.

- **`void updateSectionCellData(String originalValue, String effectiveValue, int cellVersion, String cellIdInExcelFormat)`**:
מעדכנת את הנתוני התא שנבחר בUI ומציגה אותם למשתמש

- **`void updateVersionsData(ObservableList<Integer> versionsList)`**:
מעדכנת את תיבת הגרסאות של המערכת מהם ניתן לצפות בכל אחת מהגרסאות

- **`void updateValueActionListener(ActionEvent event)`**:
מטפלת באירוע כאשר לחצן עדכון ערך התא נלחץ. מעדכנת את ערך התא שנבחר בעזרת תקשורת עם הבקר הראשי.

- **`void updateVersionsComboBoxAndLabel(boolean currentVersion)`**:
מעדכנת את החיוויים על כך שקיימת גרסה עדכנית מזו המוצגת בהתאם לפרמטר שקיבלה (שאומר האם מוצגת הגרסה העדכנית ביותר כרגע או לא).

- **<u>CommandsController CLASS</u>**
  - **תיאור:**
    מחלקה המתארת את הבקר המנהל את הפקודות הקשורות לשינויים ויזואליים של עמודות שורות ותאים כמו כן אחראית גם על פעולות הסינון והמיון של הגיליון.
  - **שיטות עיקריות**:

- **`void setMainController(MainController mainController)`**:
מגדירה את הבקר הראשי עבור בקר הפקודות, מאפשרת לו אינטראקציה עם הלוגיקה העיקרית של היישום.

- **`void initialize()`**:
מאתחלת את הבקר על ידי אכלוס תיבות הבחירה והגדרת מאזיני אירועים עבור לחצנים שונים.

- **`void showFirstSortPopup()`**:
מציגה את החלון הראשון למיון, בו המשתמש יכול להזין את הקואורדינטות של הפינה העליונה השמאלית והפינה התחתונה הימנית של הטווח שיש למיין.

- **`void showSecondSortPopup(Coordinate topLeft, Coordinate bottomRight)`**:
מציגה את החלון השני למיון, בו המשתמש יכול לבחור עמודות למיון.

- **`void showFirstFilterPopup()`**:
מציגה את החלון הראשון לסינון, בו המשתמש יכול להזין את הקואורדינטות של הפינה העליונה השמאלית והפינה התחתונה הימנית של הטווח שיש לסנן.

- **`void showSecondFilterPopup(Coordinate topLeft, Coordinate bottomRight)`**:
מציגה את החלון ההשני לסינון, בו המשתמש יכול לבחור עמודה וערכים ייחודיים לסינון.

- **`void internalhandleCellClick(Label cellLabel, int row, int col)`**:
מטפלת באירוע כאשר תא נלחץ ומעדכנת את ממשק המשתמש עם המאפיינים של התא הנבחר.

- **`void applyCellChanges()`**:
מיישמת שינויים על התא הנבחר, כמו צבע רקע וצבע טקסט.

- **`void resetCellVisual(ActionEvent event)`**:
מנקה את המאפיינים הוויזואליים של התא הנבחר לערכי ברירת המחדל שלהם.

- **`void applyColumnChanges()`**:
מיישמת שינויים על העמודה הנבחרת, כמו יישור ורוחב.

- **`void applyRowChanges()`**:
מיישמת שינויים על השורה הנבחרת, כמו גובה.

- **<u>RangesCotroller CLASS</u>**
  - **תיאור:**
    מחלקה המתארת את הבקר המנהל את הפקודות הקשורות לטווחים מאפשרת למשתמשים ליצור, להסיר ולבחור טווחים, ומתקשרת עם הבקר הראשי על מנת לבצע את הפעולות הללו.
  - **שיטות עיקריות**:

- **`void setMainController(MainController mainController)`**:
מגדירה את הבקר הראשי עבור בקר הטווחים, מאפשרת לו אינטראקציה עם הלוגיקה העיקרית של היישום.

- **`void initialize()`**:
מאתחלת את הבקר על ידי הגדרת מאזין לשינויים בבחירת תיבת השילוב של הטווחים.

- **void checkAddNewRangeActionListener(ActionEvent event):**
מטפלת באירוע כאשר תיבת הסימון "הוסף טווח חדש" נבחרת או מבוטלת. מפעילה או משביתה את שדות הקלט והכפתור ליצירת טווח חדש.

- **void createRangeActionListener(ActionEvent event):**
מטפלת באירוע כאשר לחצן "צור טווח חדש" נלחץ. ממנפת את יצירת הטווח החדש לבקר הראשי.

- **void removeRangeActionListener(ActionEvent event):**
מטפלת באירוע כאשר לחצן "הסר טווח" נלחץ. ממנפת את הסרת הטווח הנבחר לבקר הראשי.

- **void onComboBoxSelectionChanged(ObservableValue<? extends String> observable, String oldValue, String newValue):**
מטפלת באירוע כאשר הפריט שנבחר בתיבת השילוב של הטווחים משתנה. מודיעה לבקר הראשי על שינוי הבחירה בטווח.

- **void updateRanges(ObservableList<String> ranges):**
מעדכנת את תיבת הבחירה של הטווחים עם רשימת הטווחים המסופקת.

- **void activateSection():**
מאפשרת אינטראקציה של המשתמש, מפעילה את חלק הטווחים בממשק המשתמש.

- **CellSelectionPopup CLASS**
  - **תיאור:**
    מחלקה האחראית על חלון בחירת הפרמטרים עבור הslider של הניתוח הדינאמי.
    .
  - **שיטות עיקריות:**

- `openCellSelectionPopup()`:

פותחת חלון קופץ להצגת ממשק לבחירת תא והגדרת ערכי הslider.

- `areSpinnersCorrect()`:

בודקת את תקינות הערכים שהוזנו על ידי המשתמש.

- **UserLogicMapPullTask CLASS**
  - **תיאור:**
    מחלקה המרחיבה את TimerTask ומאגדת בתוכה את לוגיקת ה- pulling המבוצעת מול השרת, שולחת אליו בקשה כל פרק זמן ומקבלת ממנו את מצב כל הגיליונות במערכת בצורה של מבנה הנתונים:Map<String, Map<String,Logic>>

ומעבירה אותו לבקר הdashboard המנהל את עדכון המידע אצל הלקוח.

- **שיטות עיקריות**:

- `run()`:

מטודה אשר בפועל מבצעת את שליחת הבקשה לשרת ואחראית על לקבל ממנו את התשובה

- **LoginController CLASS**

  - **תיאור:**
    מחלקה המתארת את הבקר המנהל את ההתחברות הראשונית למערכת מאפשר למשתמש להקליד את שמו ולהתחבר למערכת בעת ההתחברות שולחת בקשה לשרת אשר מעדכן את נתוני המשתמש אצלו.

  - **שיטות עיקריות**:

- **loginButtonClicked(ActionEvent event):**
מטפלת בפעולת כפתור ההתחברות, שולחת בקשת התחברות לשרת ומטפלת בתשובה.

- **quitButtonClicked(ActionEvent e):**
סוגרת את האפליקציה.

- **switchToDashBoard(String UserName):**

מטפלת במעבר למסך הdashboard במקרה של התחברות מוצלחת לשרת.

- **Main CLASS**

  - **תיאור:**
    מחלקה המשמשת כנקודת הכניסה למערכת מרחיבה את המחלקה Application ויוצרת את הבמה והסצנה הראשית.

  - **שיטות עיקריות**:

- `void start(Stage primaryStage)`:
מתודה זאת נקראת כאשר היישום מופעל היא קובעת את הכותרת של הבמה הראשית טוענת את קובץ ה FXML עבור בקר ההתחברות למערכת(LoginController) יוצרת איתו סצנה ומציגה את הבמה הראשית(את המסך הראשי של המערכת)

*<u>**הערה:**</u> במודול הclient עבור כל אחת מהמחלקות הנ"ל(חוץ מלMAIN) קיימת מחלקה נוספת האחראית על אותו בקר בחלון אשר נפתח כאשר המשתמש רוצה לסנן/למיין או לבצע ניתוח דינאמי על גיליון ההבדל בין המחלקות הוא בעובדה שחלון זה אינו מתעדכן בזמן אמת מול השרת והעבודה מולו היא לוקאלית כלומר השרת מחזיר את הגיליון לאחר ביצוע הפעולה למשתמש הזה בלבד ולא מעדכן את שאר המשתמשים על הפעולות שבוצעו בחלון זה על הגיליון המוצג בו, פעולות שכוללות סינון/מיון/ניתוח דינאמי.

<u>מודול engine</u>

מחלקות עיקריות:

מודול זה מכיל את המימושים של כל הממשקים המוצגים במודול הDTO ובנוסף גם את המחלקות הבאות:

- **FunctionParser ENUM**
  - **תיאור**:
    ENUM המאפשר לתרגם את המחרוזת המכילה ביטוי כלשהו לסוג הביטוי המתאים, הממשק מכיל כמה ערכים המתארים כל סוג ביטוי אפשרי במערכת, מכיל מטודה סטטית אשר לוקחת ביטוי ובהתאם לצורתו שולחת אותו למטודה היעודית של כל ביטוי היודעת לפרק את המחרוזת של הביטוי הספציפי.
  - **שיטות עיקריות**:

- **public abstract Expression parse(List<String> arguments, Set<Coordinate> tempDependentCoordinates)**

כל מופע של הENUM מממש את המטודה הנ"ל בעצמו בהתאם לסוג הביטוי שלו ושולף מתוך המחרוזת שהתקבלה את הביטוי שלו.

- **public static Expression parseExpression(String input, Set<Coordinate> tempDependentCoordinates)**

המטודה בודקת את המחרוזת שהתקבלה ויודעת בהתאם לסוגה להעביר אותה למטודה parse המתאימה.

  - **ערכי הFunctionParser במערכת:**

- ***IDENTITY***
- ***PLUS***
- ***MINUS***
- ***UPPER_CASE***
- ***REF***
- ***TIMES***
- ***DIVIDE***
- ***MOD***

- ***POW***
- ***ABS***
- ***CONCAT***
- ***SUB***
- ***SUM***
- ***AVERAGE***
- ***PERCENT***
- ***EQUAL***
- ***NOT***
- ***BIGGER***
- ***LESS***
- ***OR***
- ***AND***
- ***IF***

- **LogicManager CLASS**
  - **תיאור**:
    מחלקה המכילה את מבנה הנתונים הראשי של המערכת מבנה נתונים המכיל את כל גליונות המערכת ושמות כל המשתמשים שהעלו אותם ומאגדת את כל הפעולות אשר ניתן לבצע עליו . מבנה נתונים מהצורה: Map<String, Map<String,Logic>>
  - **שיטות עיקריות**:

- `synchronized createUsersLogicList(String username):`

מטודה שיוצרת עבור משתמש מפה חדשה אשר תכיל את הגיליונות שיעלה בעתיד למערכת

- `addLogicToUser(String username, Logic logic):`

מוסיפה גיליון חדש עבור המשתמש במפה שלו

- **SheetManager CLASS**
  - **תיאור**:
    מחלקה המכילה את שמות כל הגיליונות במערכת ואחראית לעובדה שיהיה רק שם ייחודי לכל גיליון במערכת.
  - **שיטות עיקריות**:

- `void addNameToSheetManager(String SheetName):`

מוסיפה גיליון חדש למאגר השמות בתנאי שהוא לא קיים כבר (ההוספה והבדיקה מבוצעת בבלוק מסונכרן על מנת שכמה משתמשים יוכלו להוסיף גיליונות במקביל)

- **UserManager CLASS**
  - **תיאור**:
    מחלקה המכילה את שמות כל המשתמשים במערכת ואחראית לעובדה שיהיה שם ייחודי לכל משתמש במערכת.
  - **שיטות עיקריות**:

• `synchronized void addUser(String username)`
מתודה מסונכרנת שמוסיפה משתמש חדש עם שם המשתמש שניתן כפרמטר למערכת.

• `boolean isUserExists(String username)`
מתודה שבודקת אם קיים משתמש במערכת עם שם המשתמש שניתן כפרמטר ומחזירה ערך בוליאני בהתאם.
(false אם המשתמש קיים, אחרת true)

מודול serializer

מודול זה מכיל את הserializers הבאים המשמשים להעברת הנתונים בין השרת ללקוח:

Serializers:

- **CellSerializer**
- **CoordinateSerializer**
- **EffectiveValueSerializer**
- **PermissionRequestSerializer**
- **ReadOnlyCellSerializer**
- **SheetSerializer**
- **LogicSerializer**
- **SheetSerializer**

Deserializers:

- **CellDeserializer**
- **CoordinateDeserializer**
- **EffectiveValueDeserializer**
- **LogicDeserializer**

- **PermissionRequestDeserializer**
- **ReadOnlyCellDeserializer**
- **ReadOnlySheetDeserializer**
- **SheetDeserializer**

- **<u>מודול webServer</u>**

מודול זה הוא בעצם הwebModule של המערכת ומכיל את הservlets הבאים:

Servlets:

- AckOrDenyPermissionServlet
- ChangePermissionServlet
- DynamicAnalysisServlet
- FileUploadServlet
- FilterServlet
- FirstFilterPopupCheckServlet
- FirstSortPopupCheckServlet
- LightweightLoginServlet
- PermissionRequestServlet
- RangesServlet
- SortServlet
- UpdateCellServlet
- UserLogicMapServlet

בנוסף להם מכיל גם את הממשק הבא:

- **<u>ExceptionHandler INTERFACE</u>**
  - **תיאור:**
    ממשק המכיל מטודות סטטיות המחזירות הודעה ייעודית למשתמש עבור כל exception שנזרק מהמנוע.
  - **שגיאות אליהם הממשק נותן הודעה ייעודית:**

- CoordinateOutOfRangeException
- CoordinateParamOutOfRangeException
- FileTypeIsntSupportedException
- FunctionDoesNotExistException
- InvalidExcelFormatException
- CircularDependencyException
- StartIndexLowerThenEndException
- RangeDoesNotExistException
- RangeAlreadyExistsException

- RangeCoordinateOutOfRangeException
- RangeInUseException
- RangeStartIndexLowerThenEndException
- RangeContainsNoAppropriateCellsException
- FilterSortRangeStartIndexLowerThenEndException

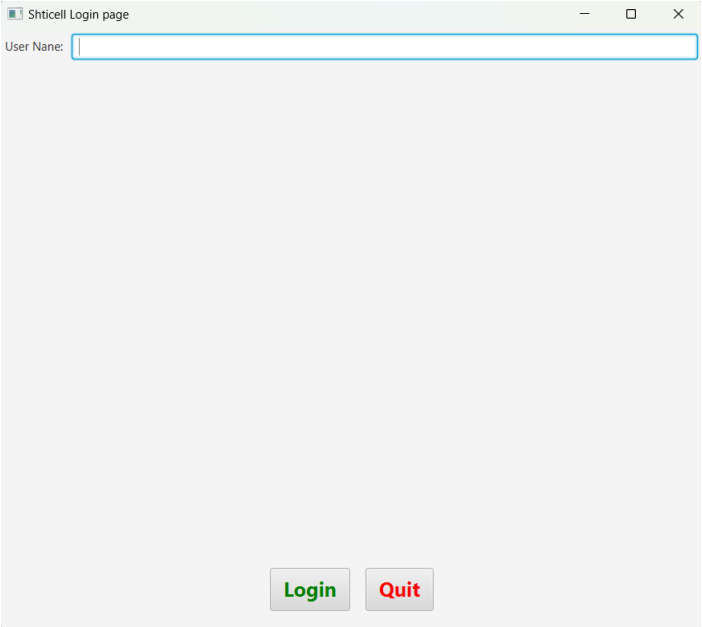
מסכים עיקריים:

מסך התחברות למערכת:

- **תיאור:**

מדובר במסך הראשי של האפליקציה המוצג למשתמש עם הפעלת האפליקציה, על מנת להתחבר למערכת יש להקליד שם משתמש וללחוץ על LOGIN:

- **צילום מסך:**

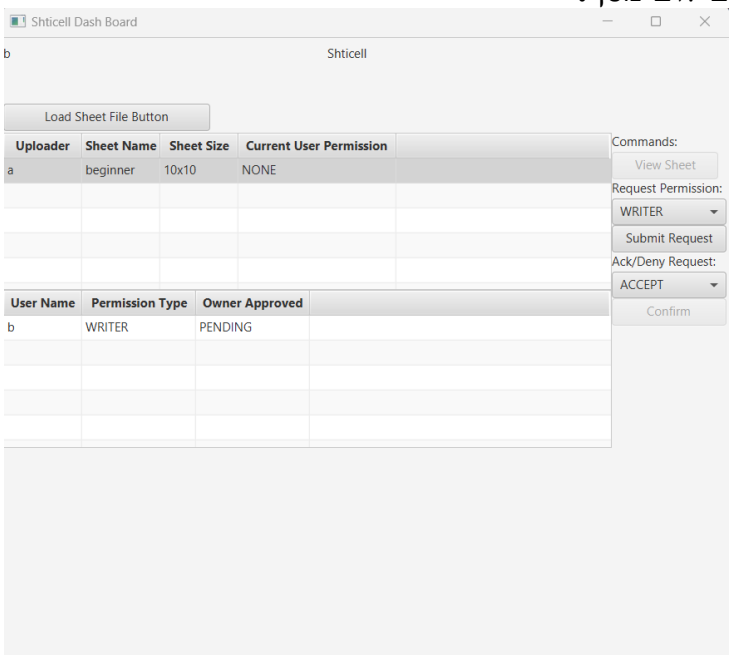


מסך DASHBOARD:

- **תיאור:**

מדובר במסך המוצג למשתמש לאחר התחברות מוצלחת בו הוא יכול לראות את כלל הגיליונות הקיימים במערכת ואת ההרשאות שלו אליהם, לבקש הרשאה לגיליונות שאינם שלו, לאשר בקשות עבור גיליונות שלו ולצפות/לערוך גיליון בהתאם להרשאותיו אליו.

- **צילום מסך:**

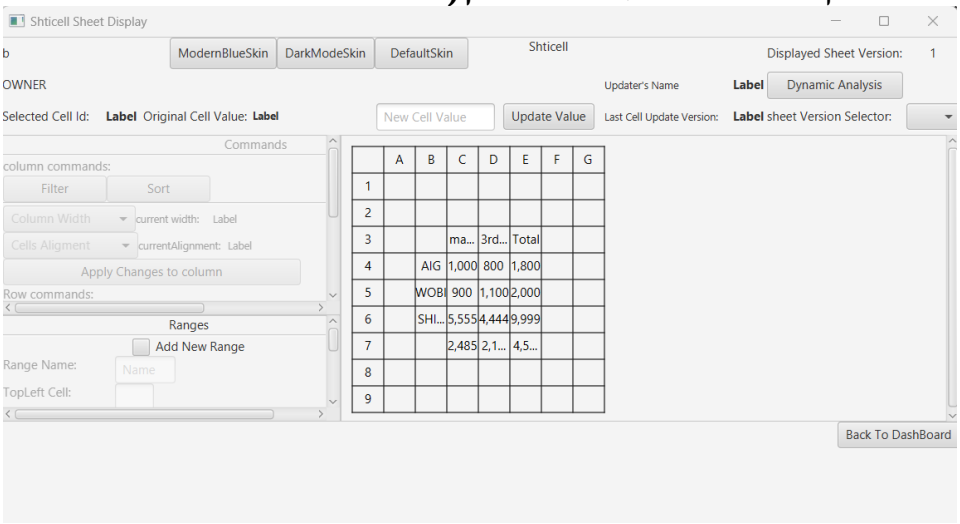


מסך תפעול גיליון:

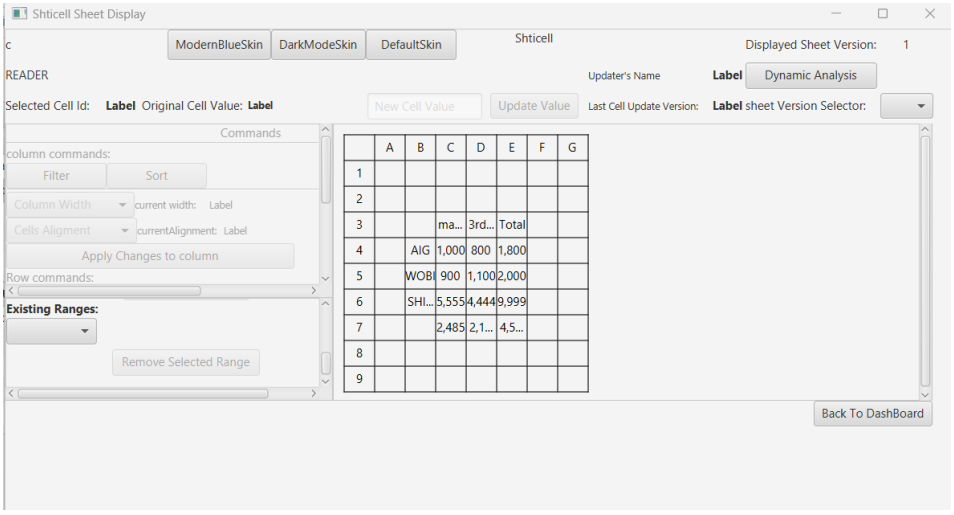
- **תיאור:**

מדובר במסך אשר נפתח לאחר לחיצת המשתמש על view sheet עבור גיליון מסוים אליו יש לו לפחות הרשאת קריאה. במסך זה ניתן לערוך/לצפות בגיליון בהתאם להרשאות המשתמש לאחר לחיצה על תא מסוים יפתחו חלקים מסויימים בהתאם להרשאות המשתמש:

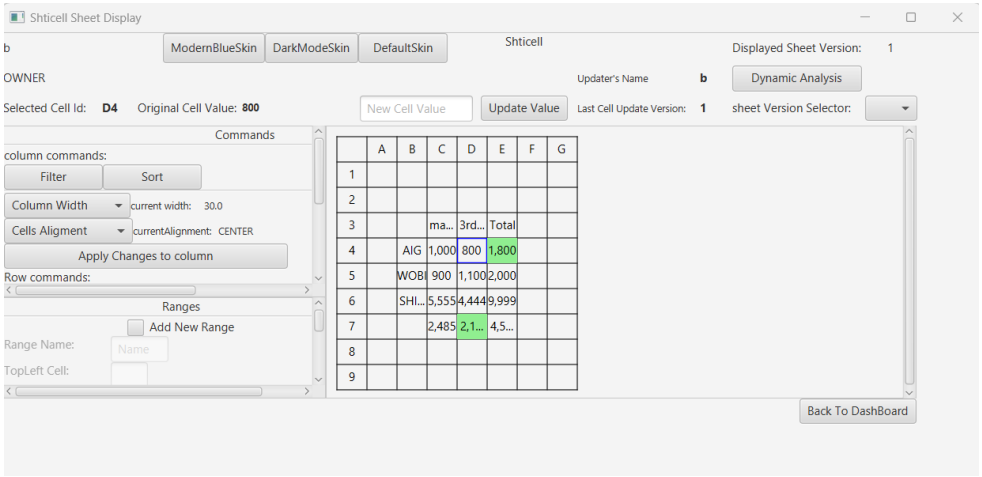
- **צילום מסך לפני לחיצה על תא בגיליון(הרשאות WRITER/OWNER):**



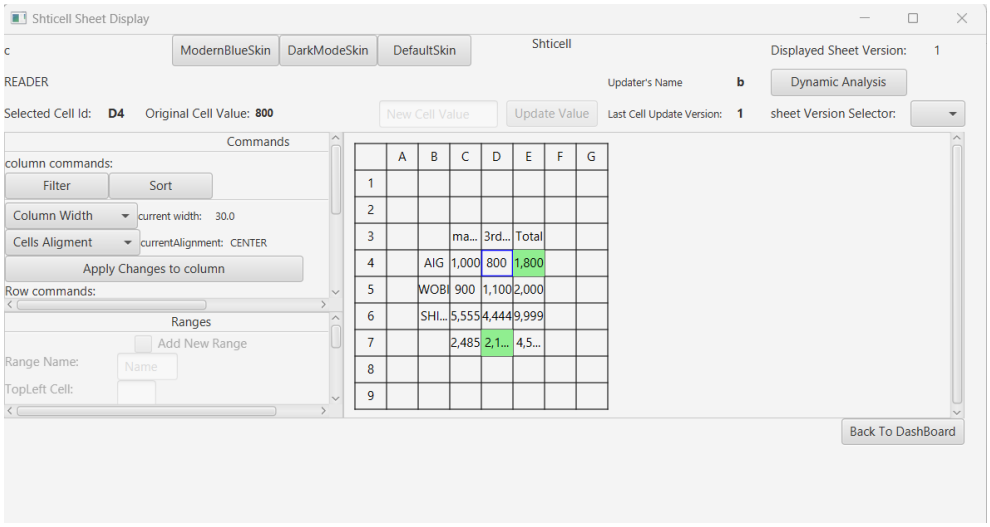
- **צילום מסך לפני לחיצה על תא בגיליון(הרשאות READER):**



- **צילום מסך לאחר לחיצה על תא בגיליון(הרשאות WRITER/OWNER):**



- **צילום מסך לאחר לחיצה על תא בגיליון(הרשאות READER):**

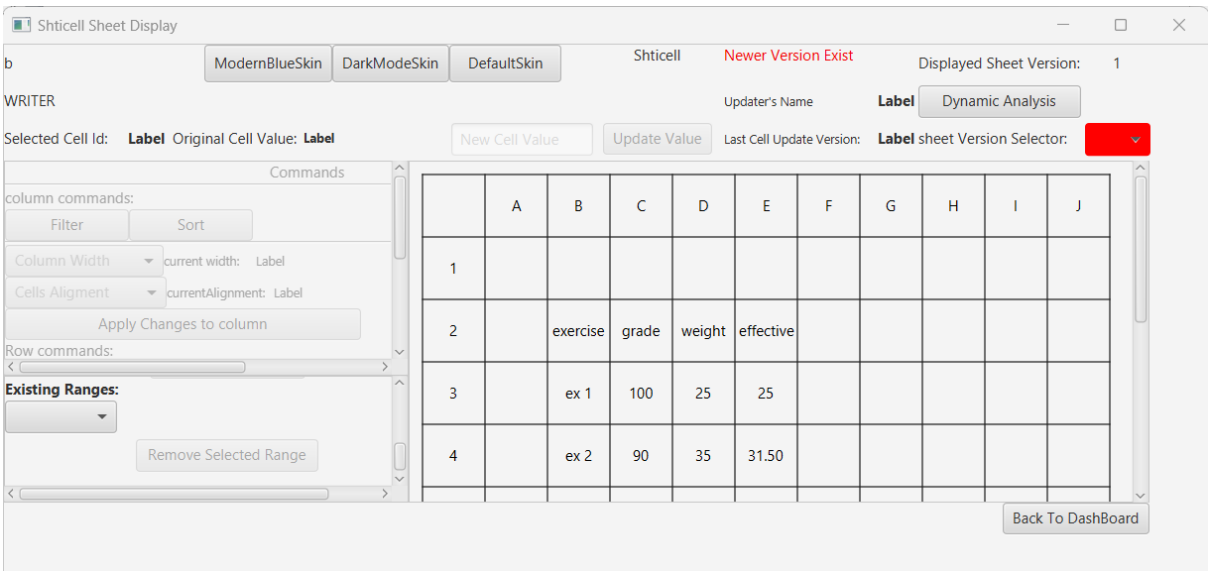


חיווי עבור גרסה לא עדכנית של גיליון:

- **תיאור:**

במידה ומשתמש כרגע צופה בגרסה לא עדכנית של הגיליון מהסיבה שמשתמש אחר עדכן אותה תוך כדי. המשתמש שאינו עדכן יקבל את החיווי הבא:

- **צילום מסך:**

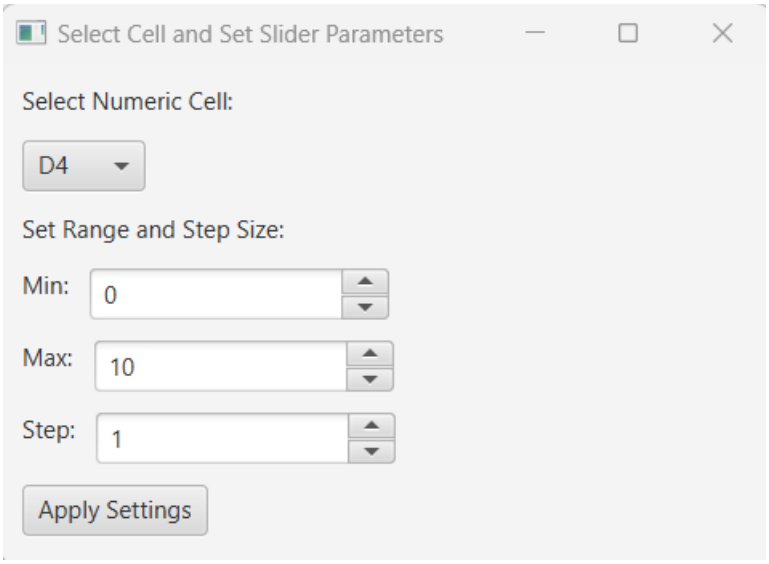


מסך בחירת פרמטרים לSlider של ניתוח דינאמי:

- **תיאור:**

מדובר במסך אשר נפתח לאחר לחיצת המשתמש על כפתור dynamic analysis במסך תפעול גיליון ומאפשר לבחור את הפרמטרים עבור הslider אשר ישמש אותנו בניתוח הדינאמי.

- **צילום מסך:**



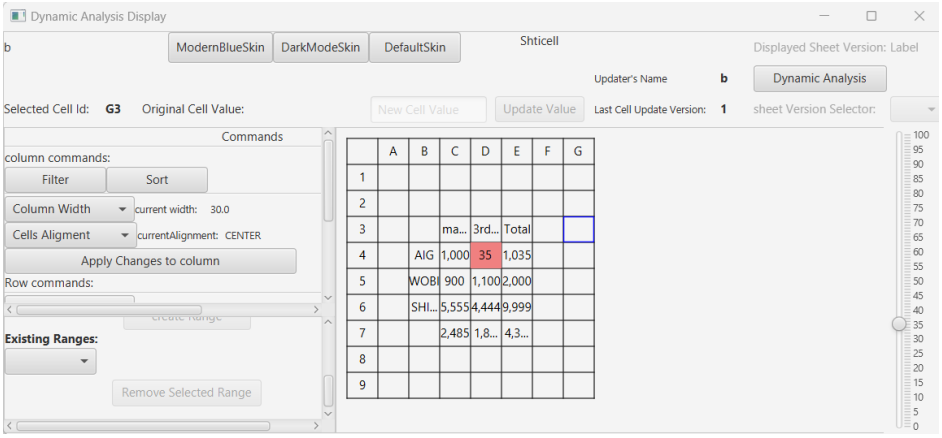
מסך ניתוח דינאמי:

- **תיאור:**

מדובר במסך אשר נפתח לאחר בחירת הפרמטרים עבור הslider. במסך זה ניתן לבצע ניתוח או פילטור/מיון כמו כן, ניתן לראות כי התא אשר עליו עושים כרגע את הניתוח הדינאמי מסומן באדום וניתן גם להחליפו ע" לחיצה נוספת על כפתור ה dynamic analysis.

(זה גם המסך שנפתח כאשר המשתמש רוצה למיין/לסנן גיליון)

- **צילום מסך:**



חלון שני של מסך מיון גיליון:

- **תיאור:**

מדובר בחלון המיון שנפתח לאחר בחירת הטווח אותו רוצים למיין. באזור הבחירה למעלה מוצגות העמודות הקיימות באזור שהמשתמש בחר למיין, יש לבחור את העמודות שתרצה למיין לפיהם, העמודות שבחרת יוצגו באזור הבחירה התחתון לפי הסדר שבחרת בהם מלמעלה למטה וזה יהיה גם סדר המיון לפי העמודות.
(בחירה מרובה של עמודות מתבצעת ע" החזקת CTRL תוך כדי הלחיצה עם העכבר על העמודות אותן תרצה לבחור).

- **צילום מסך:**