**ספר פרויקט - ענן**



**שם: אילן ישוק**

**ת"ז: 326102126**

**תאריך הגשה: 15 ביוני, 2022**

**מנחה: ברכה עסיס**

**חלופה: תכנון ותכנות מערכות מידע**

**שם הפרויקט: Safe Cloud**

**בי"ס: מקיף אמי"ת באר שבע**

**תוכן עניינים**

**מבוא**

**תיאור תכולת הפרויקט**

מסמך זה מאפיין, מפרט ומציג את התוצר הסופי אותו אני מגיש במסגרת פרויקט הגמר במגמת הנדסת תוכנה בכיתה י"ב. בנוסף לתוצר וליכולות שלו, הוא מציג אתגרים, דרכי התמודדות עמם, הסברים לדברים עקיפים בנושא ועוד מגוון רב של תכנים. מסמך זה מהווה את כל מה שצריך לדעת על הפרויקט, ויותר מכך.

**מוטיבציה לפרויקט**

כולם מכירים את גוגל דרייב. זו אחת מהתוכנות המשומשות בעולם. כל כך משומשת, עד שהיא שינתה את הדרך שבה אנו מעבירים קבצים ממחשב למחשב. אמנם גוגל דרייב מצויין בכל מידותיו, אבל אני תמיד רציתי משהו פשוט יותר, מהיר יותר, שניתן לבצע פעולות כמו הוספת/מחיקת/הורדת קבצים בלחיצה אחת.

אני לא מתיימר להיות גוגל דרייב או איי-קלאוד הבא ולא מנסה להתחרות בעשרות אלפי אנשים, אבל אני כן מציג דרך נוספת למשתמש, פשוטה ומהירה לשמירת קבצים בענן, בצורה הכי מאובטחת שיש (אם ירצה בזאת).

**הפרויקט**

הפרויקט עצמו הוא ענן, ששומר את הקבצים שלך במאגר נתונים הרחק ממך, ובצורה מאובטחת אם תבחר בכך, בשביל שתוכל להשתמש בקבצים האלו במקומות אחרים, במחשבים אחרים ובזמנים אחרים. אני ניסיתי בכוונה לעשות את הממשק הגרפי פשוט ומהיר ככל הניתן, אני לא אוהב להתעסק עם כל הפונקציונליות (שלדעתי מיותרת) בהרבה מהעננים שיש כיום ולכן ניסיתי לצמצם ככל הניתן במישור הזה.

בשביל להשתמש בענן אתה צריך משתמש. לאחר הרשמה שכוללת אימות דו-שלבי עם האימייל, אתה תגיע לאתר שיכיל את הקבצים שלך. אתה תוכל להוסיף קבצים ע"י גרירתם למסך, למחוק ולהוריד בלחיצה. ניתן להוסיף כמות רבה של קבצים בגרירה אחת. בנוסף, אתה יכול לבחור להוסיף ולהוריד קבצים בעזרת מפתח מיוחד. על המפתח אפרט בהמשך, אבל אציין שבעת הכנסת המפתח, המשתמש מוסיף עוד שכבת הגנה – קריטית וטובה מאוד – לקבציו המועברים באינטרנט אל הענן. חוץ מעמוד הקבצים, יש עמוד פרופיל שמציג את פרטיך ונותן לך להתנתק מהמשתמש אם תרצה ועמוד הסבר על הפונקציונליות של התוכנה.

השרת שלי זוכר משתמשים נכנסים ויוצאים, בצורה בטוחה, אצלו בדיסק. תודות לכך, משתמש לא צריך להכנס כל פעם מחדש לאתר, ולמלא את פרטיו. האתר זוכר אותו למשך זמן של יום אחד (שיש שיגידו קצר ויש שיגידו ארוך, אני מאמין שיום אחד זה כמות זמן טובה לנוחות וטובה לאבטחה). כמובן שהשרת לא יודע לא את סיסמת המשתמש, ואם המשתמש יבחר – גם לא את קבצי המשתמש.

**אתגרים שפגשתי במהלך הפרויקט**

פגשתי מס' לא מבוטל של אתגרים בזמן עשיית הפרויקט והנה כמה מהם:

- **אין זמן**. כיתה י"ב עמוסה גם ככה, בגרויות מפיזיקה, מתמטיקה, תנ"ך ולכן היה צריך לנהל את הזמן בצורה יעילה, יחד עם הצבת יעדים לסיום חלקים בעבודה.

- **ממשק גרפי**. כמעט ואף פעם לא עבדתי עם ממשק גרפי כמו שעבדתי בעבודה זו. הממשק הוא אתר אינטרנט, ולכן יצא לי לעבוד עם javascript הרבה, הרבה יותר ממה שידעתי או הכרתי. למדתי מהבסיס ביותר - כיצד javascript רץ ועד מטרותיו והדברים הקטנים והנוחים שהופכים את השפה הזו לכל כך משומשת בעולם.

- **תכנות אסינכרוני** - הרבה בעיות נוצרו עקב זה שjavascript היא שפה שעובדת רק בצורה אסינכרונית. כלומר, היה צריך לשנות את התפיסה שלי - שכל שורה שאני כותב, אני צריך להוסיף לה תכנותית, "לחכות" שהשורה תסתיים ורק אז לשלוח את זה. זה כולל שימוש בPromiseים, async וכו'.

- **אבטחה**. ניסיתי לשים דגש על האבטחה משתי סיבות. ראשית, זהו פרויקט בסייבר, כלומר צריך לגעת באבטחת מידע ורשתות. שנית, בן אדם שירצה לשלוח את הקבצים האלו, ירצה ברוב המקרים, שהם יישארו רק שלו ואף אחד, כולל אני (או מנהל מאגר הנתונים) לא ידע מה יש שם. האתגר הוא - כיצד לעשות את זה? מאיפה מתחילים בכלל?

- **הקלאסי**. לשבת על בעיה כמה שעות, שאין לך שמץ של מושג למה משהו לא עובד בכלל. אתן דוגמה על מה אני מדבר. כשאני מוסיף משתמש למאגר נתונים, אני עושה גיבוב של חלק מהנתונים שלו, בצורה כזו שיהיה קשה מאוד (כל כך שאפשר להגיד בלתי אפשרי) לחזור לנתוניו המקוריים (מהמחרוזת המגובבת). הבעיה שלי הייתה שמשום מה זה לא היה קבוע. בסופו של דבר, אחרי המון בדיקות הגעתי לפתרון והוא להוסיף סוגריים ל datetime.timestamp משום שזו פעולה. נשמע טיפשי נכון? אבל אז קפצה בעיה אחרת. עדיין הגיבוב לא היה קבוע. הפעם גיליתי, שאני שומר תאריך יצירת משתמש מדויק - עד לשניות במאגר הנתונים אבל כשהייתי עושה את הגיבוב (לשם השוואה לאחר הכנסת משתמש נכנס), התאריך היה מדויק לרמת המיקרו-שניות, ולכן בעת הוצאה ממאגר הנתונים וגיבוב המידע, לא הייתי מקבל שווי בין נתונים למרות שהיו שווים. בקיצור.... הקלאסי. וזה קרה פעמים רבות, אבל זה התהליך לתוצר הסופי וכמות הלמידה שניתן להוציא מנסיונות כאלו היא ללא הגבלה.

**קהל יעד**

קהל היעד של הפרויקט הוא בערך כל מי שמשתמש היום באינטרנט - כלומר כל אחד שרוצה את הקבצים שלו ביותר ממקום אחד. אנשים שעובדים עם חומר וצריכים אותו מסודר - רופאים, מורים, הייטק, פקידים, תלמידים ועוד...

**פיתוחים עתידיים**

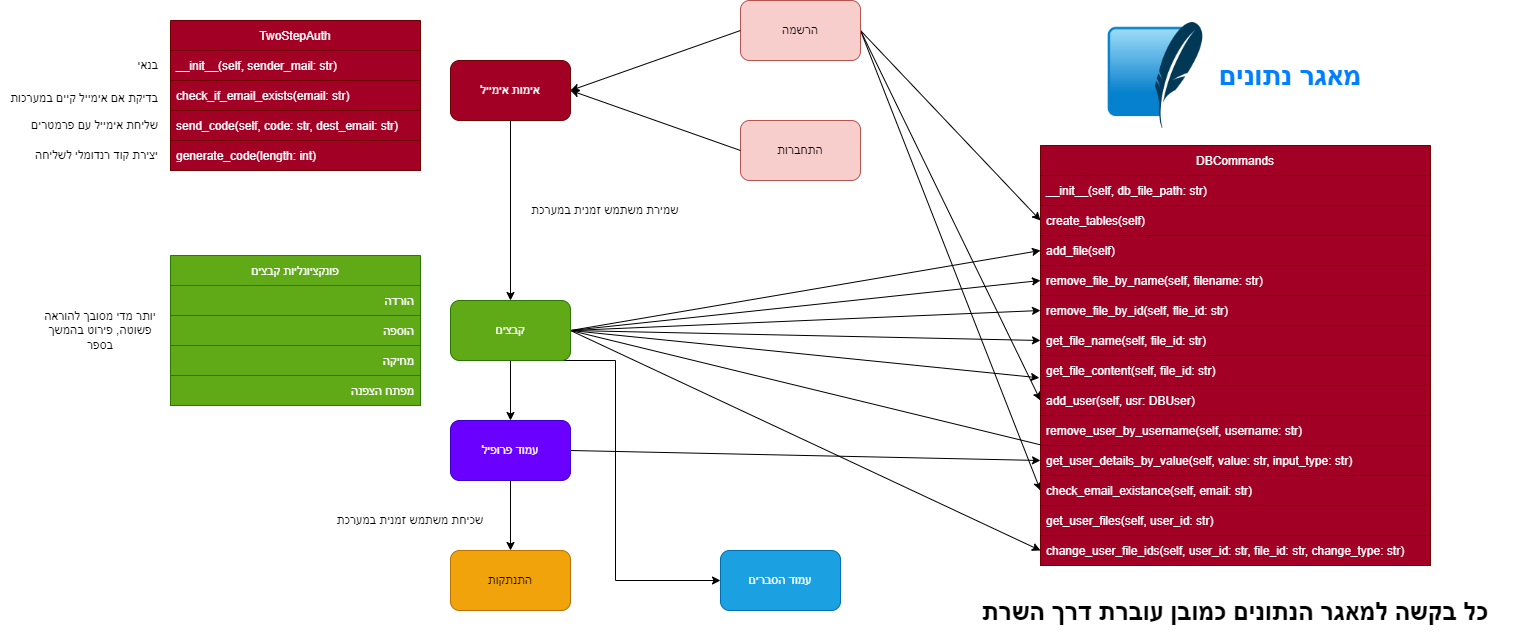
כמעט כל מה שתכננתי לפרויקט, בוצע. לכן, אין הרבה פערים שנוצרו אבל יש דברים שהיה נחמד להוסיף:

- שיתוף קבצים בין כמה אנשים, כלומר קישור קובץ מסוים ליותר מבן אדם אחד

- יצירת תיקיות שיכילו קבצים (כמובן לא באופן "זריקת תיקייה"... כי בדקתי... לא כל כך אפשרי)

**ארכיטקטורת הפרויקט**

ניתן לייצג את ארכיטקטורת הפרויקט באמצעות התרשים הבא:



**להוסיף דיאגרמה של endpoints ופעולות שמזומנות מהפלאסק -> פעולות שרת**

ניתן לחלק את הפרויקט לכמה מישורי עבודה שונים:

- **מאגר נתונים**:

- טבלת משתמשים

- טבלת קבצים

- **אבטחה**:

- אימות משתמש דו-שלבי

- הצפנת קבצים במאגר הנתונים

- גיבוב פרטיו של המשתמש ליצירת מחרוזת מזהה ייחודית

- **ממשק גרפי**

- תהליך הרשמות משתמש חדש

- תהליך התחברות של משתמש קיים

- ניהול קבצים (הוספה/מחיקה/הורדה)

הממשק הגרפי הוא אתר, ולכן בקשות HTTP (POST, GET) מהאתר יהיו הקלטים. הבקשות יהיו הקלטים לשרת והכנסת הנתונים לאתר עצמו יהיו הקלטים של המשתמש. קלטים של שאר המישורים, אבטחה ומאגר נתונים, אלו קלטים שמספק אך ורק השרת (כמובן לפי בקשות המשתמש).

**הסדר שמוצג פה הוא גם הסדר לפיו יוצגו החומרים בספר הפרויקט, משום שבדרך זו הכי קל להבין מדוע כל דבר נעשה בצורה שנעשה.**

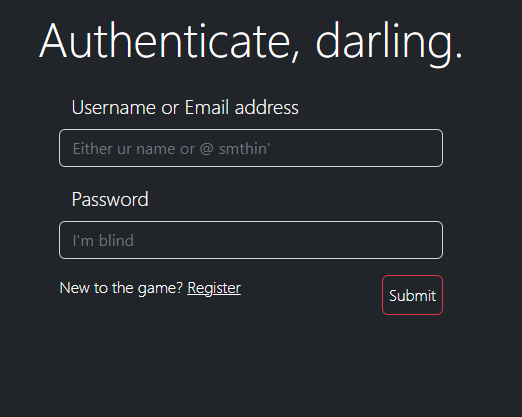
**מדריך למשתמש**

**התקנת התוכנה**

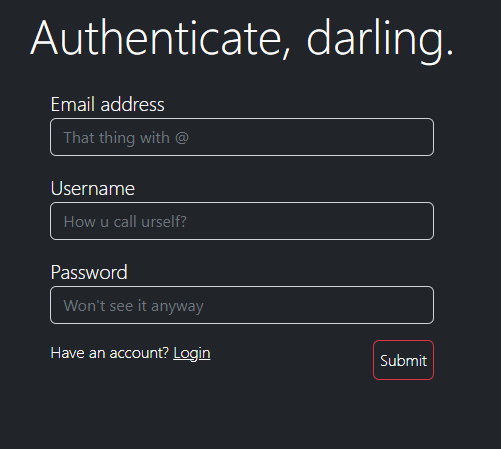
עוד לא פירטתי אבל יהיה כאן כעקרון git clone ו virtual env אז זה יהיה כזה העתקה וpip install

**הרצת התוכנה**

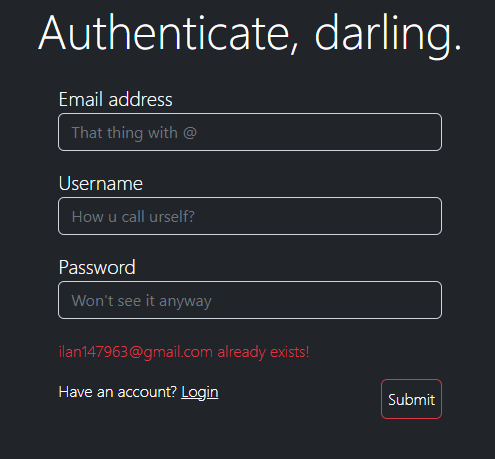
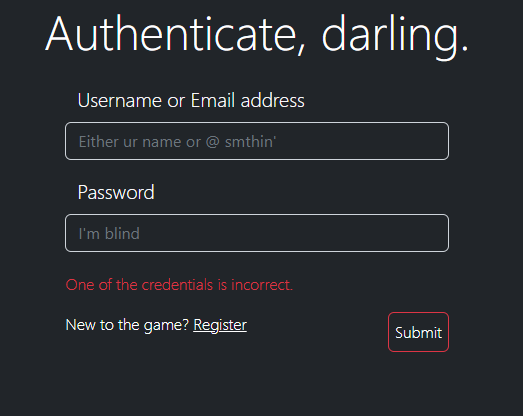
בעת פתיחת האתר שירוץ ב 127.0.0.1:5000 (אלא אם הוגדר אחרת) יפתח **חלון התחברות**.



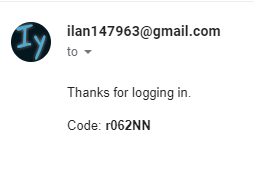
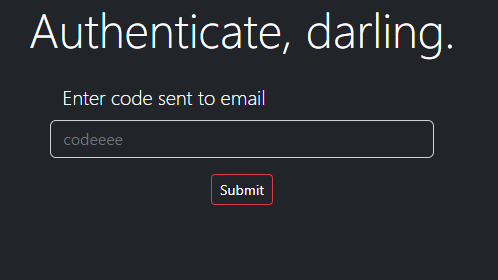
למקרה ואין משתמש, יש ללחוץ על Register בשביל לעשות משתמש חדש. בעת לחיצה **יתקבל חלון רישום משתמש חדש**:



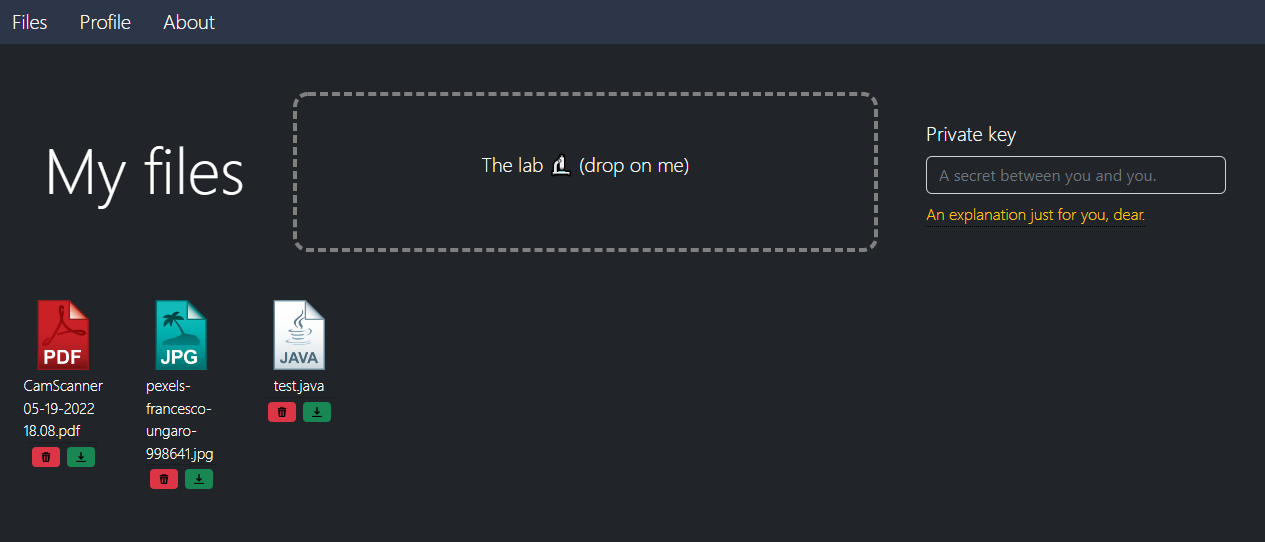
החלונות האלו נותנים פידבק למקרה ויש הכנסת נתונים תפוסה או שגויה:

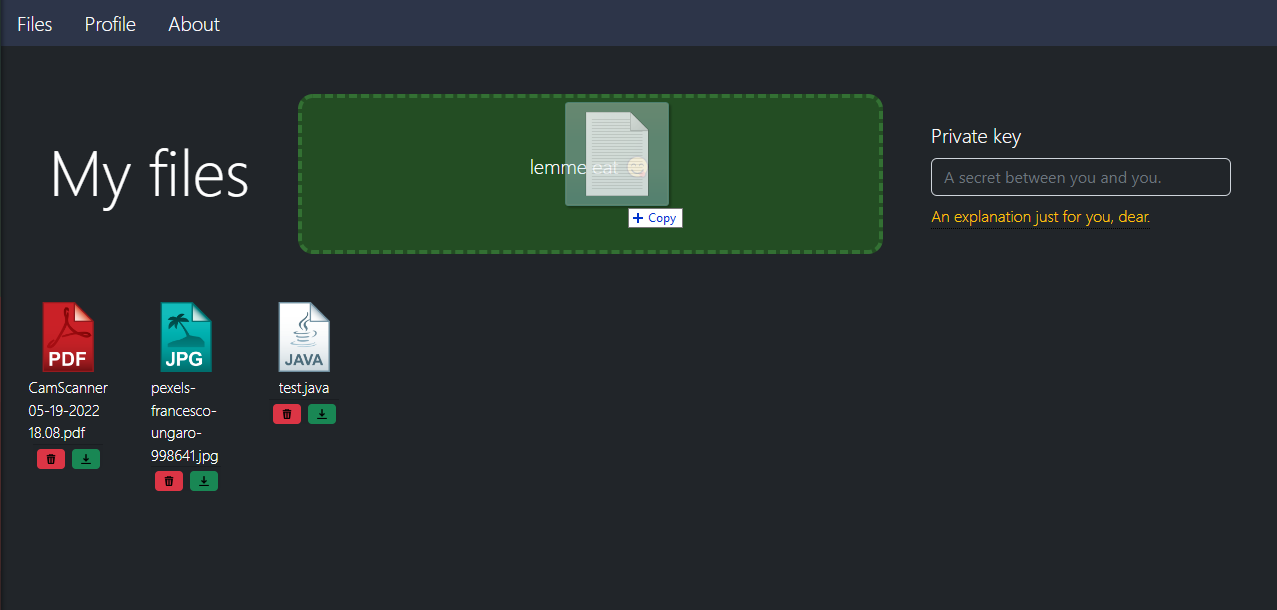
לא משנה מה תהיה הפעולה – רישום משתמש חדש או התחברות, החלון הבא שיקפוץ, לאחר לחיצה על submit הוא **חלון אימות האימייל**. החלון נועד לכך שאנשים לא סתם יכניסו אימייל שהוא לא שלהם, ושלמקרה שגילו את פרטי המשתמש שלהם, עדיין לא יוכלו להכנס למשתמש כי הפורצים לא יחזיקו בתיבת הדואר (שמאובטחת על ידי google, apple, yahoo וכו').

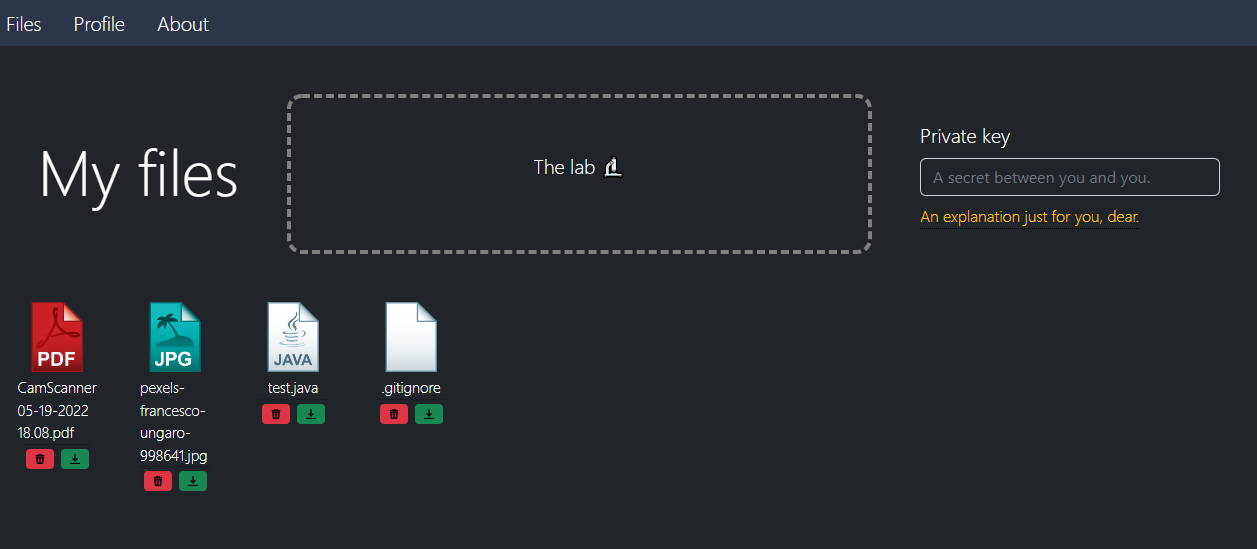
לאחר התחברות תקינה (אם תכניס קוד שגוי, תקבל פידבק על זה), מגיעים לדף העיקרי של הפרויקט והוא **דף הקבצים:**

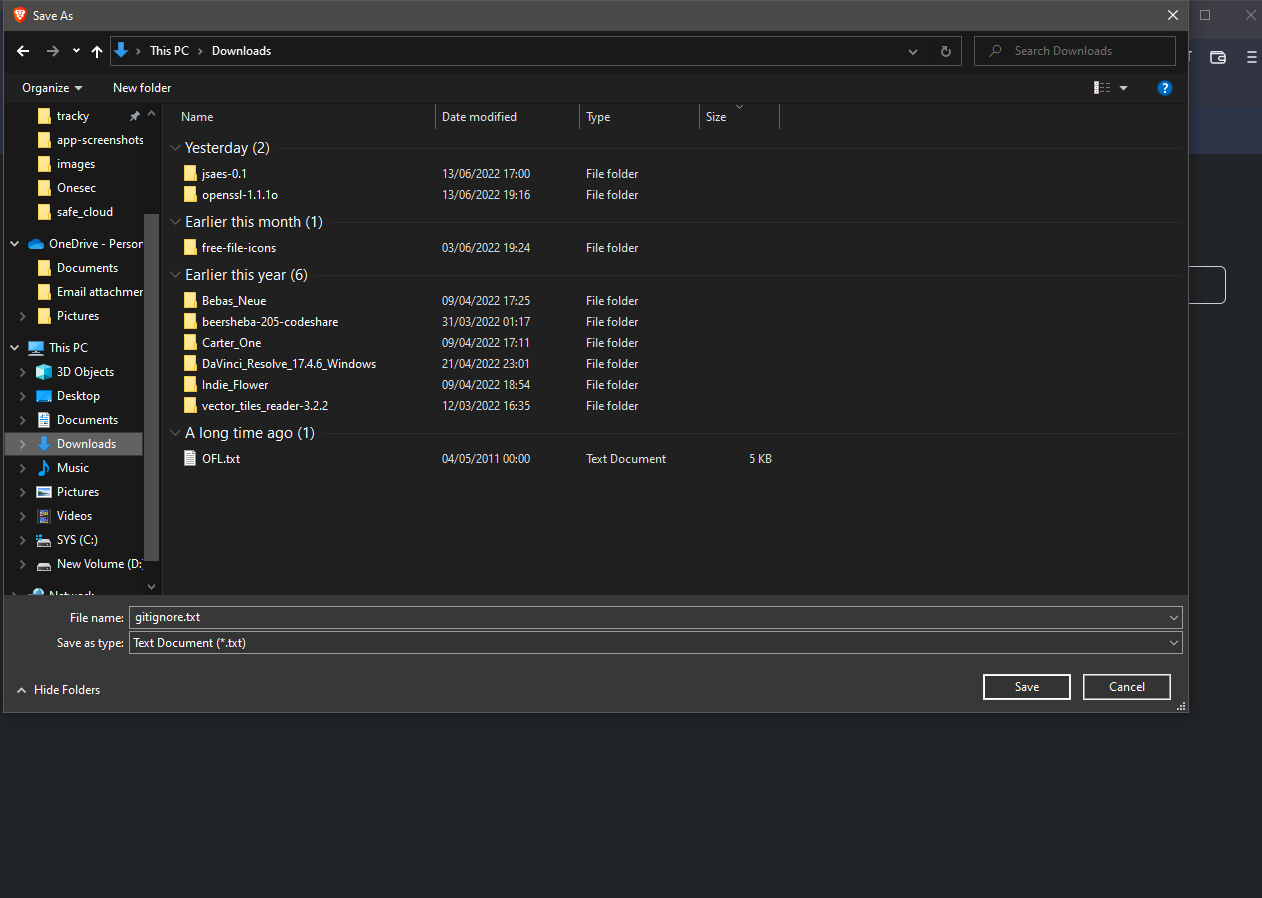


נכנסתי למשתמש שכבר יש בו קבצים, אז זה טען את הקבצים הקודמים. אם נרצה להוסיף קובץ, נגרור את הקובץ לתוך המסגרת האמצעית:

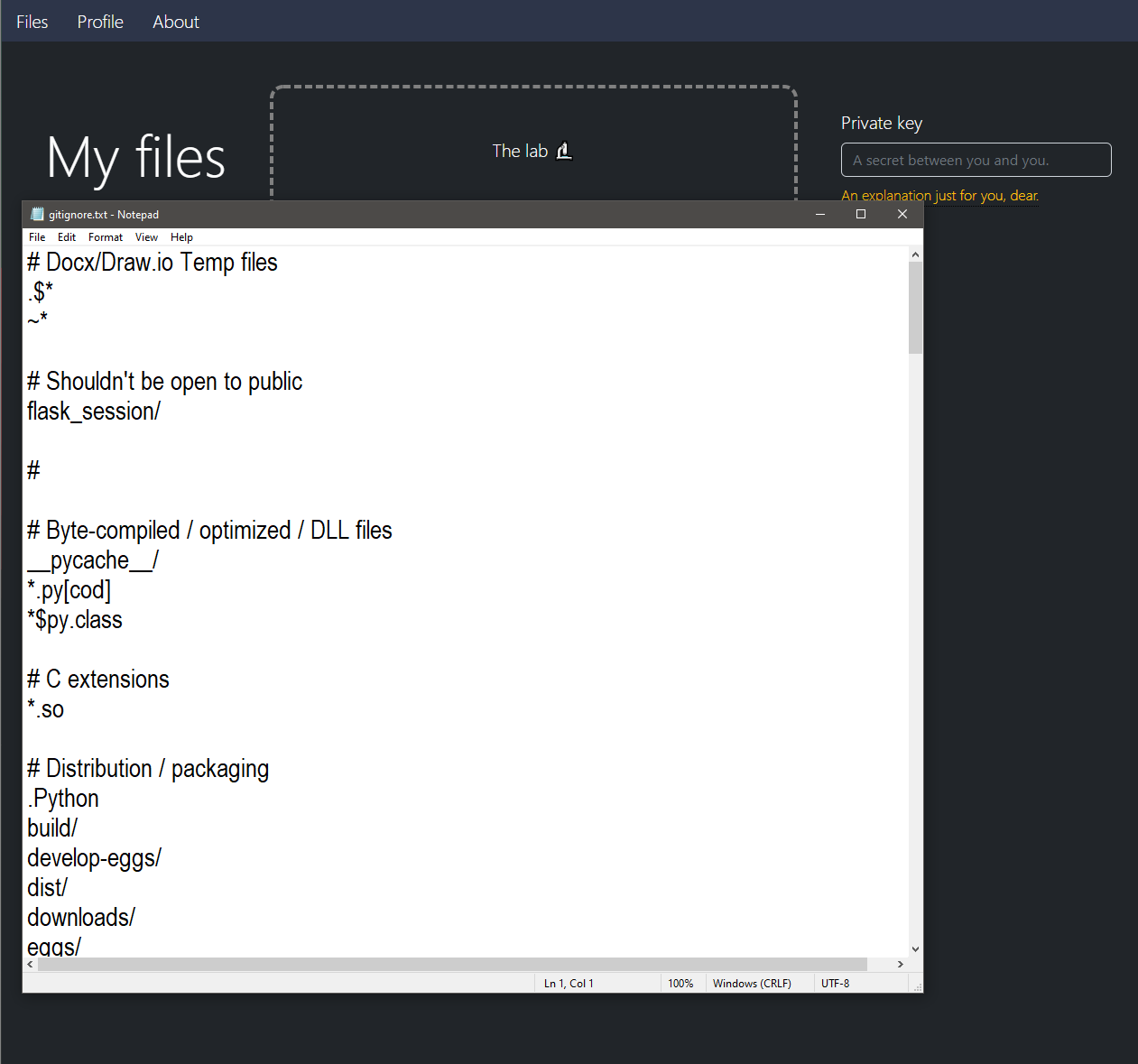


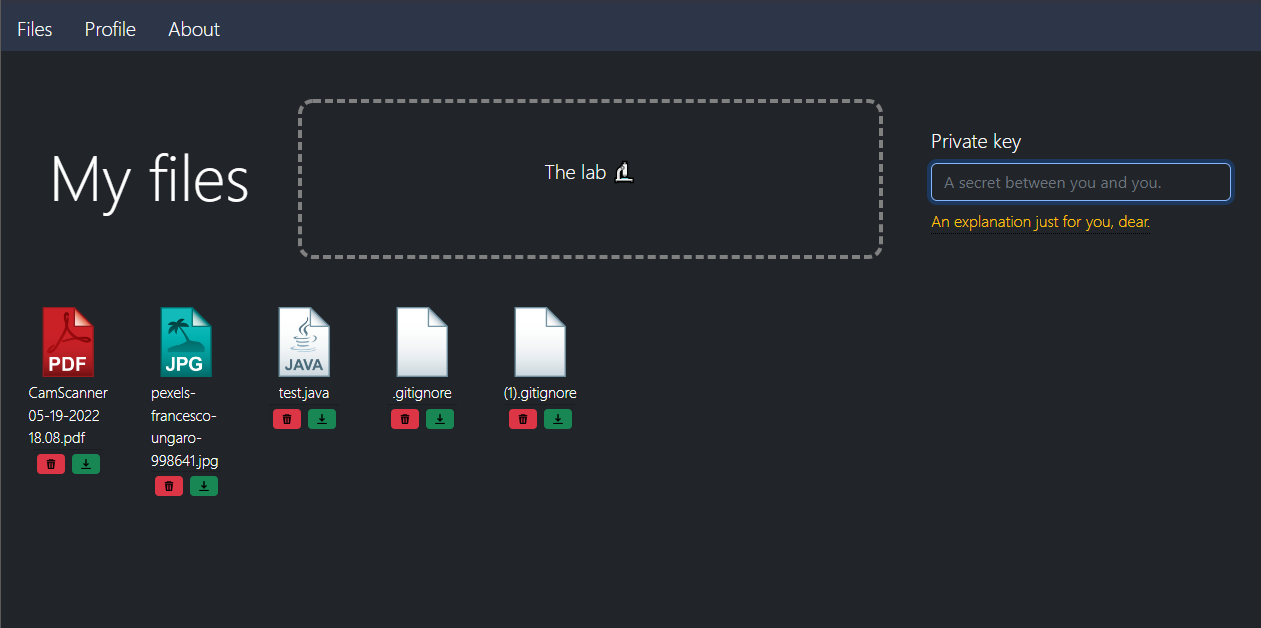
נראה שהקובץ אכן נוסף:



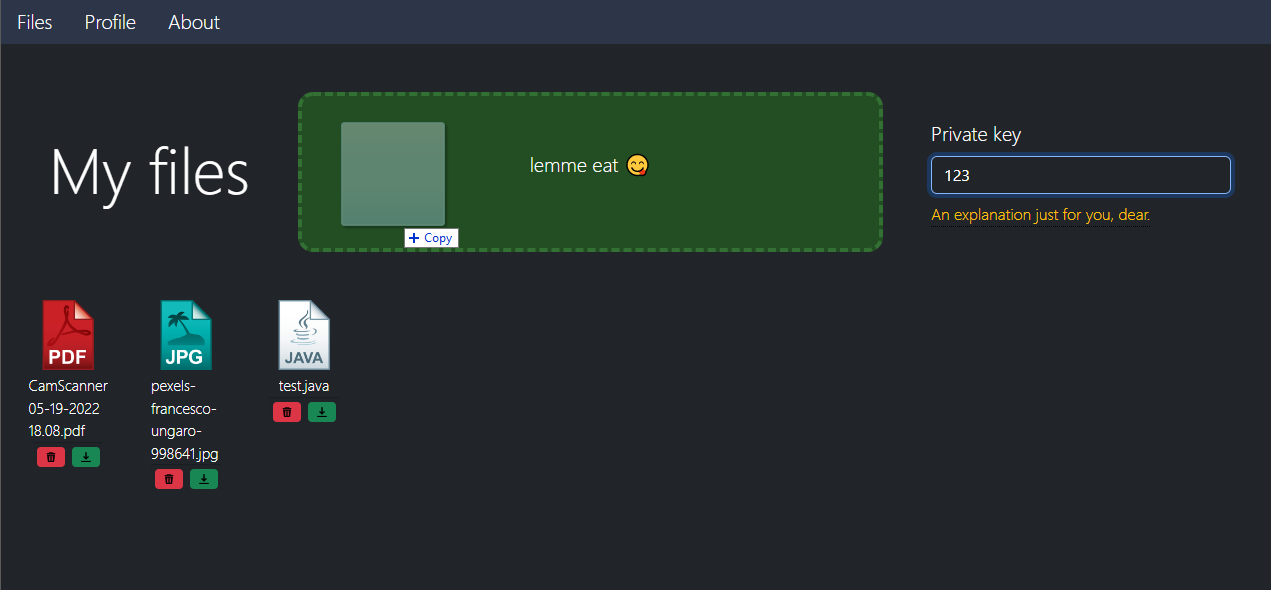
בנוסף, אם נרצה להוריד אותו, **נלחץ על הלחצן הירוק** ולמחיקה נלחץ על הלחצן האדום. בעת הורדה האתר יבקש ממנו את מיקום ההורדה: 

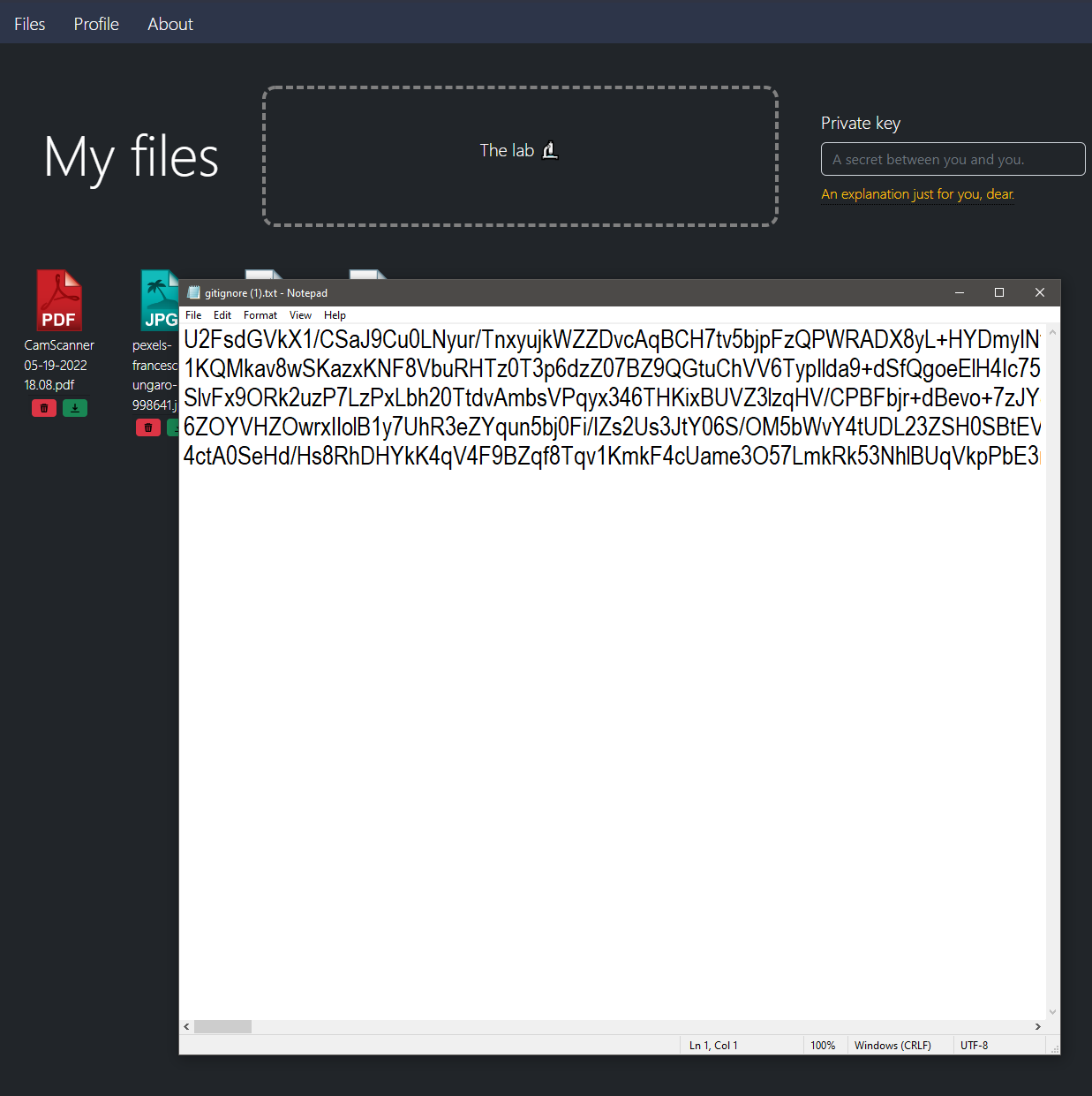
ונראה שהקובץ אכן הורד כמו שצריך.



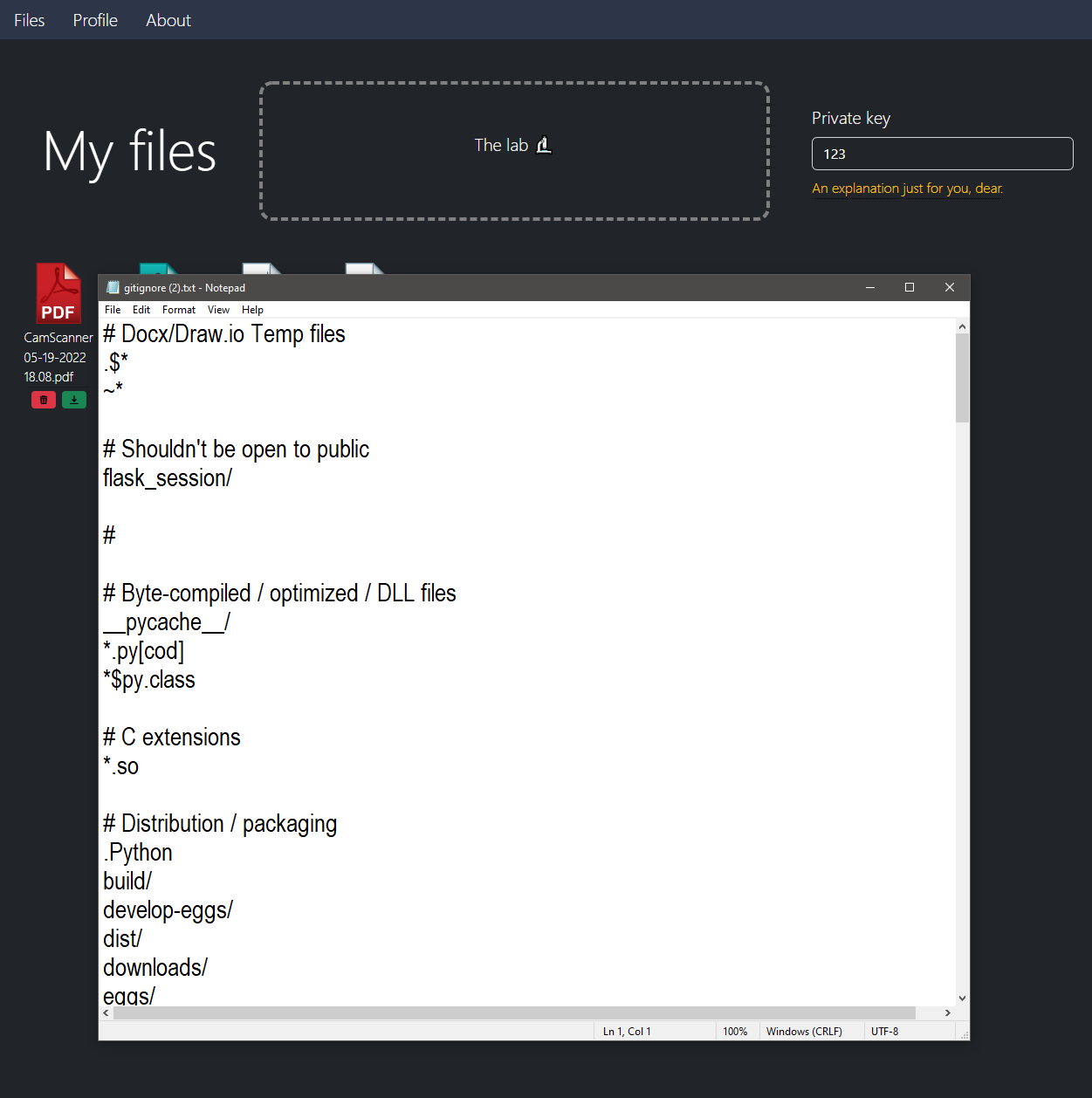
במקרה שנכניס את אותו קובץ פעמיים, נראה שהשם משתנה בהתאם: 

כמו כן, לדעתי הפיצ'ר הכי חזק בתוכנה הזאת, שאנחנו יכולים להצפין קובץ מצד הלקוח. כלומר, להצפין את הקובץ עוד לפני שיוצא מהמחשב שלנו – וכך אף אחד מלבדנו לא יוכל לדעת מה יש בקובץ, אלא אם כן הוא **מחזיק בקוד שלנו**. יש לזכור קוד הצפנה ולהוסיף קובץ כשהוא מוקלד:



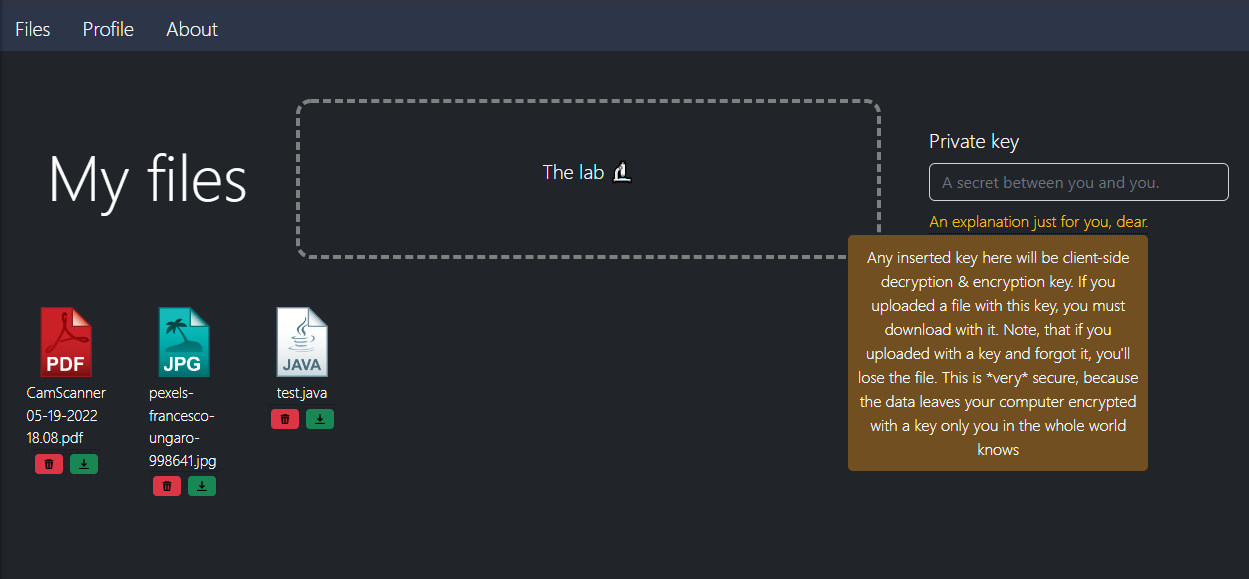
כעת, הקובץ שלנו נשלח בהצפנת AES עם המפתח "123". אם ננסה להוריד בלי המפתח: 

לא נצליח לקרוא את הקובץ. אך אם נוריד את הקובץ עם המפתח:

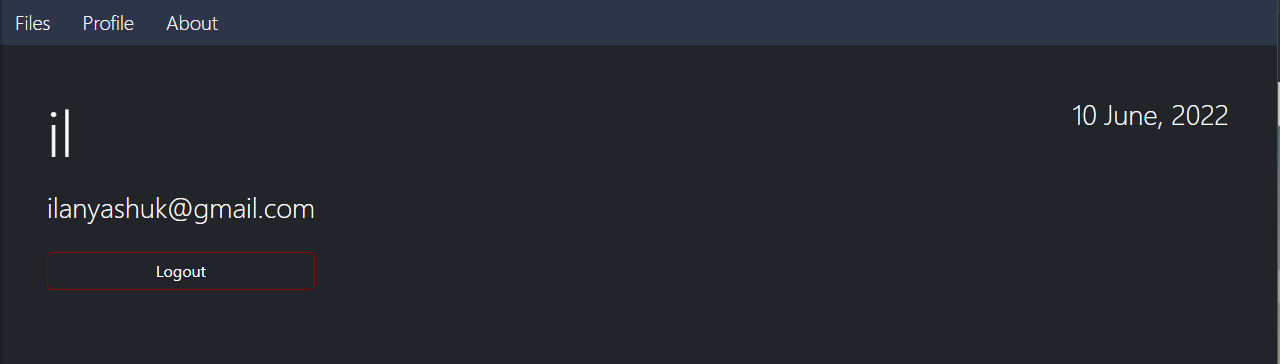


בום!

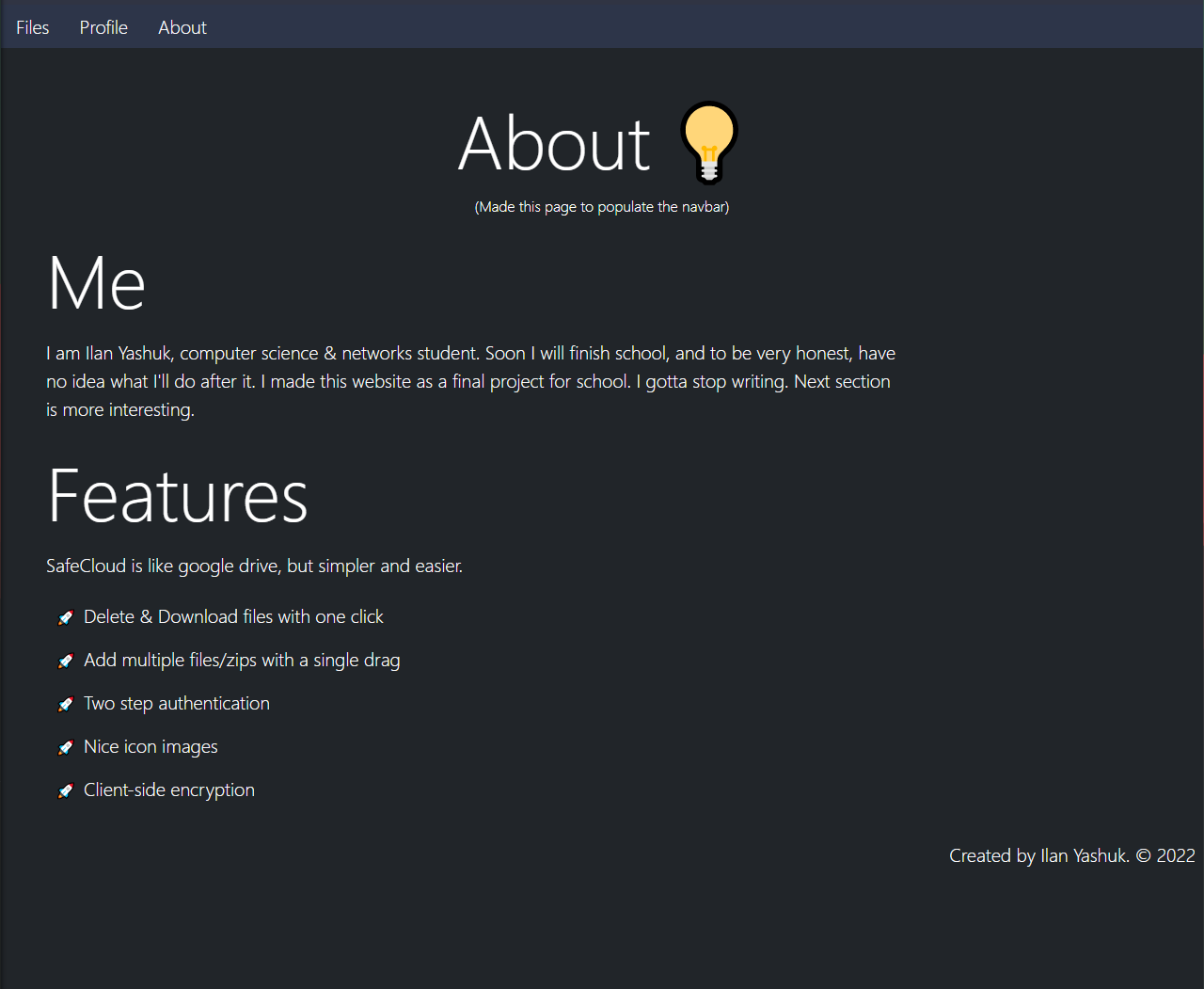
חוץ מזה, אציין שעוד פיצ'ר נחמד הוא שאפשר לגרור כמה קבצים ביחד (וזה עובד גם עם הצפנה). אם המשתמש יצטרך הסבר יותר מפורט, הוא יוכל להצביע על הטקסט הכתום:



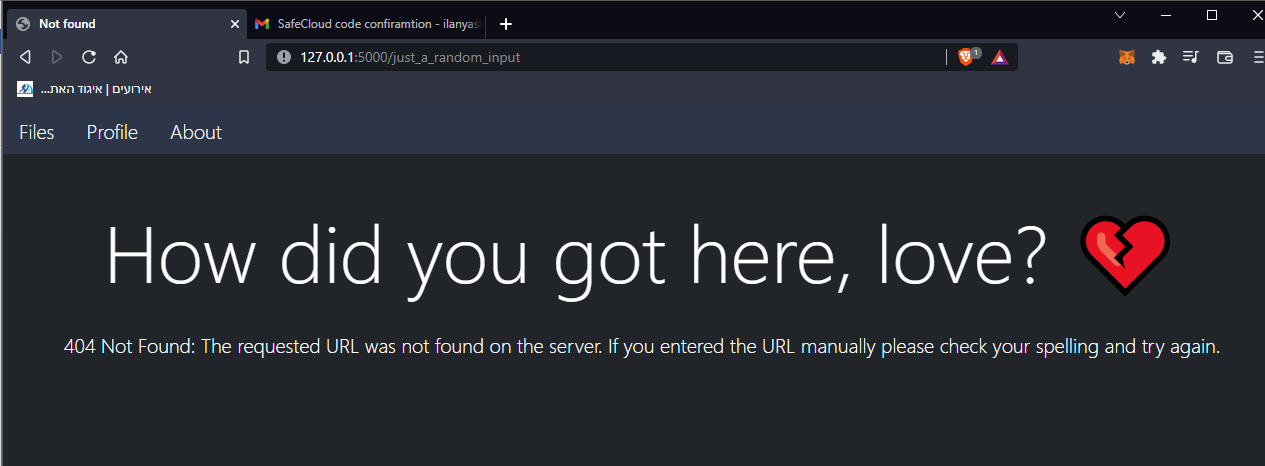
עוד עמוד מעניין הוא **עמוד הפרופיל**, שלא מכיל הרבה אבל מכיל את האפשרות להתנתק מהמשתמש. בלחיצה על הכפתור, השרת ישכח את המשתמש ואת ה "session" עמו, עד להתחברות חדשה.



שני הדפים האחרונים הם **דף שמסביר על פונקציונליות האתר:**



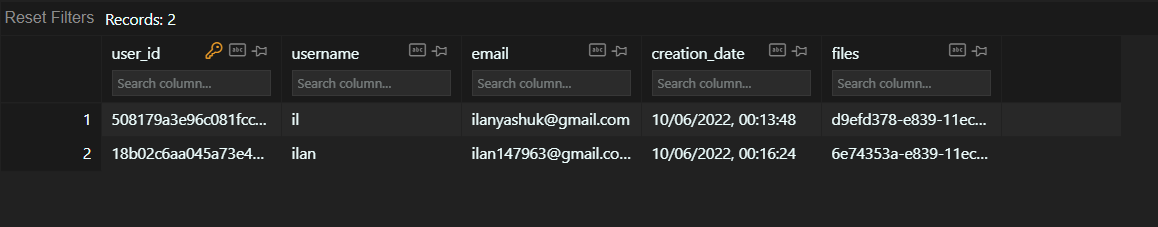
**ודף שגיאה**:



**בסיס הנתונים – sqlite3**

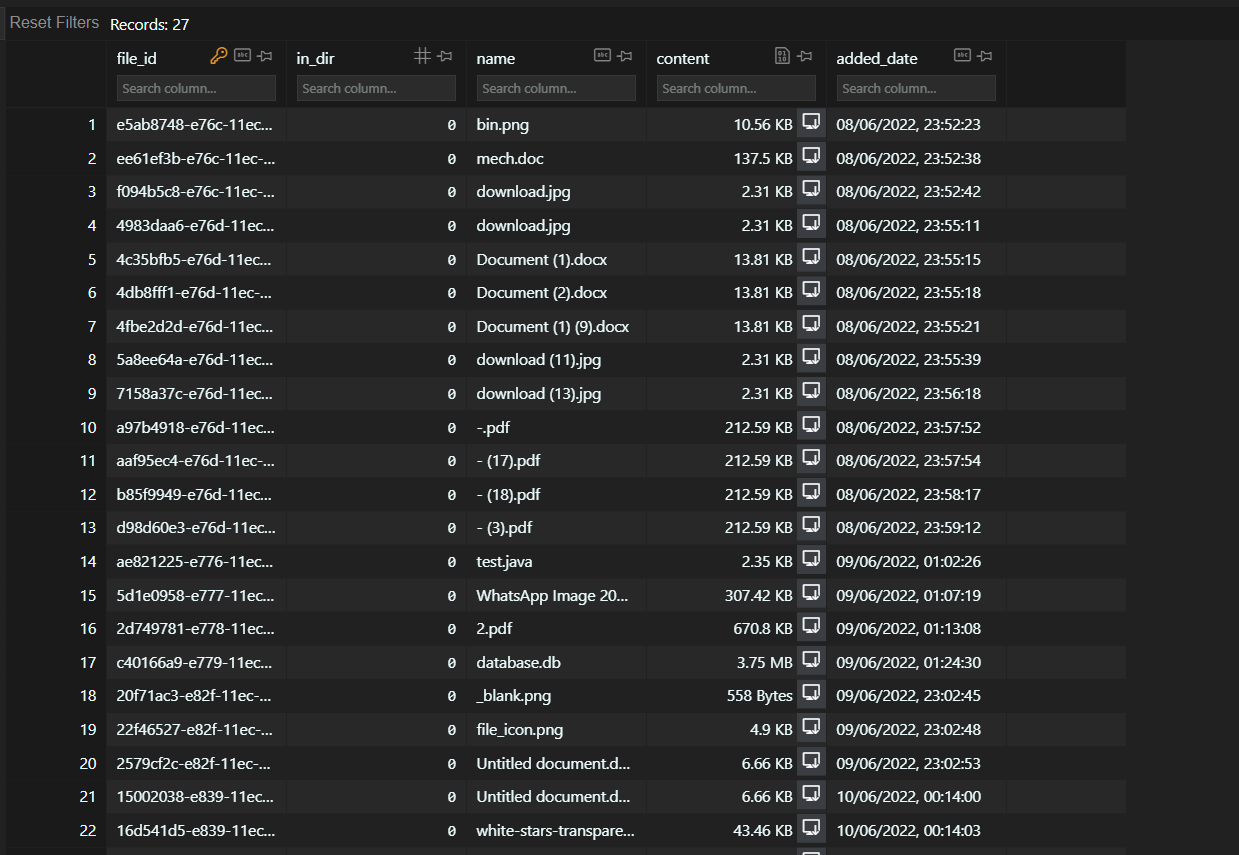
בסך הכל יש רק שתי טבלאות במאגר הנתונים. טבלת משתמשים וטבלת קבצים:

**טבלת משתמשים (דוגמה)**



* מס' מזהה למשתמש, מיוצר ע"י גיבוב סיסמה, תאריך יצירה, ואימייל, מפתח ראשי
* שם משתמש, לא מוצפן
* אימייל, גם לא יכול להיות מוצפן בשביל אימות דו-שלבי
* תאריך יצירה
* קבצים, מחרוזת המכילה את כל המס' הייחודיים של הקבצים של אותו משתמש

**טבלת קבצים (דוגמה)**



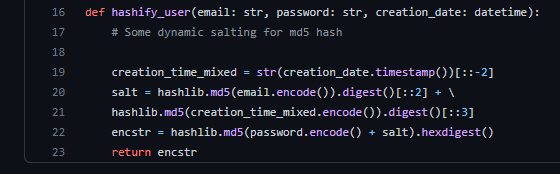
* מס' ייחודי, מבוסס זמן ולכן ייחודי לכל קובץ, מפתח ראשי
* האם בתוך תיקייה, למקרה של מימוש תיקיות בעתיד
* שם הקובץ
* תוכן הקובץ (כאן נכנסת ההצפנה אם המשתמש יבחר)
* תאריך הוספה

**אבטחה – גיבוב נתונים, אימות דו-שלבי, הצפנת צד לקוח**

לכל הקבצים הרלוונטים, ראה כותרת "המדריך למפתח". בחלק זה אציג את הדברים השונים שאני עושה למען ביטוח והצפנה של פרטי ומידע המשתמש. הכותרות מדורגות לפי רמת האבטחה, כלומר גיבוב נתונים מהווה אבטחה ברמה גבוהה, אך אין פה קונספט יוצא מן הכלל. לעומת זאת, הצפנת צד-לקוח היא מההצפנות החזקות שקיימות, לא מעצם אלגוריתם מופלא או טריק מתמטי (למרות שגם מזה אפשר להתפאר) אלא מהקונספט אותו אסביר תחת התת-כותרת המתאימה.

**גיבוב נתונים – המס' זיהוי של המשתמש**

אסור בשום פנים ואופן שהשרת, או האחראי על מבנה הנתונים, יוכל לגשת למשתמשים השמורים במאגר. דבר כזה יהיה אסון אם יתגלה על ידי המשתמשים משום שיהווה פגיעה איומה בפרטיות שלהם. לצורך מניעת בעיה כזו, אני מקבל את פרטי המשתמש, מפעיל עליהם פונקציית גיבוב, שכידוע לא ניתן לשחזרה (מכיוון שמדובר [בפונקציות מתמטיות חד-כיווניות](https://he.wikipedia.org/wiki/%D7%A4%D7%95%D7%A0%D7%A7%D7%A6%D7%99%D7%99%D7%AA_%D7%92%D7%99%D7%91%D7%95%D7%91). הנה הקוד:



* קבלת אימייל, סיסמה, תאריך יצירה כקלט
* הפיכת זמן יצירה לכמות שניות מאז 1970 ובלבול המחרוזת
* יצירת [מלח](https://infosecscout.com/md5-salt-hash/#:~:text=not%20so%20much%20%E2%80%A6-,What%20is%20MD5%20Salt%20and%20How%20to%20Use%20It%3F,to%20get%20the%20same%20result.), שהוא גיבוב מחרוזת אימייל מבולבלת בתוספת גיבוב המחרוזת המבולבלת של זמן היצירה
* גיבוב המלח בתוספת הסיסמה לקבלת התוצאה הסופית
* התוצאה היא גיבוב הנתונים, תוצאה יחידה במינה לכל משתמש, מס' זיהוי המשתמש

**אימות דו-שלבי**

לפרט

**הצפנת צד לקוח**

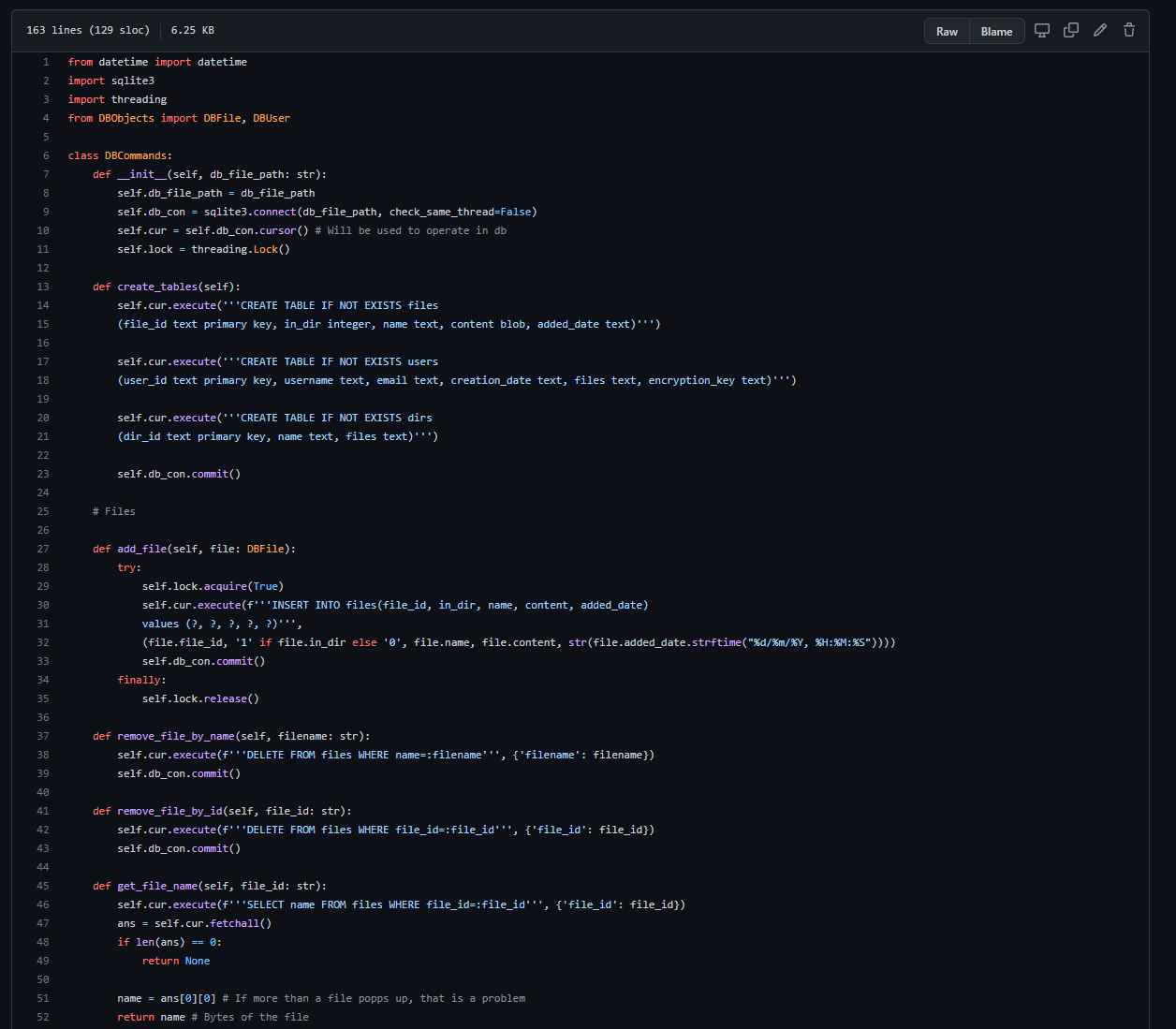
לפרט

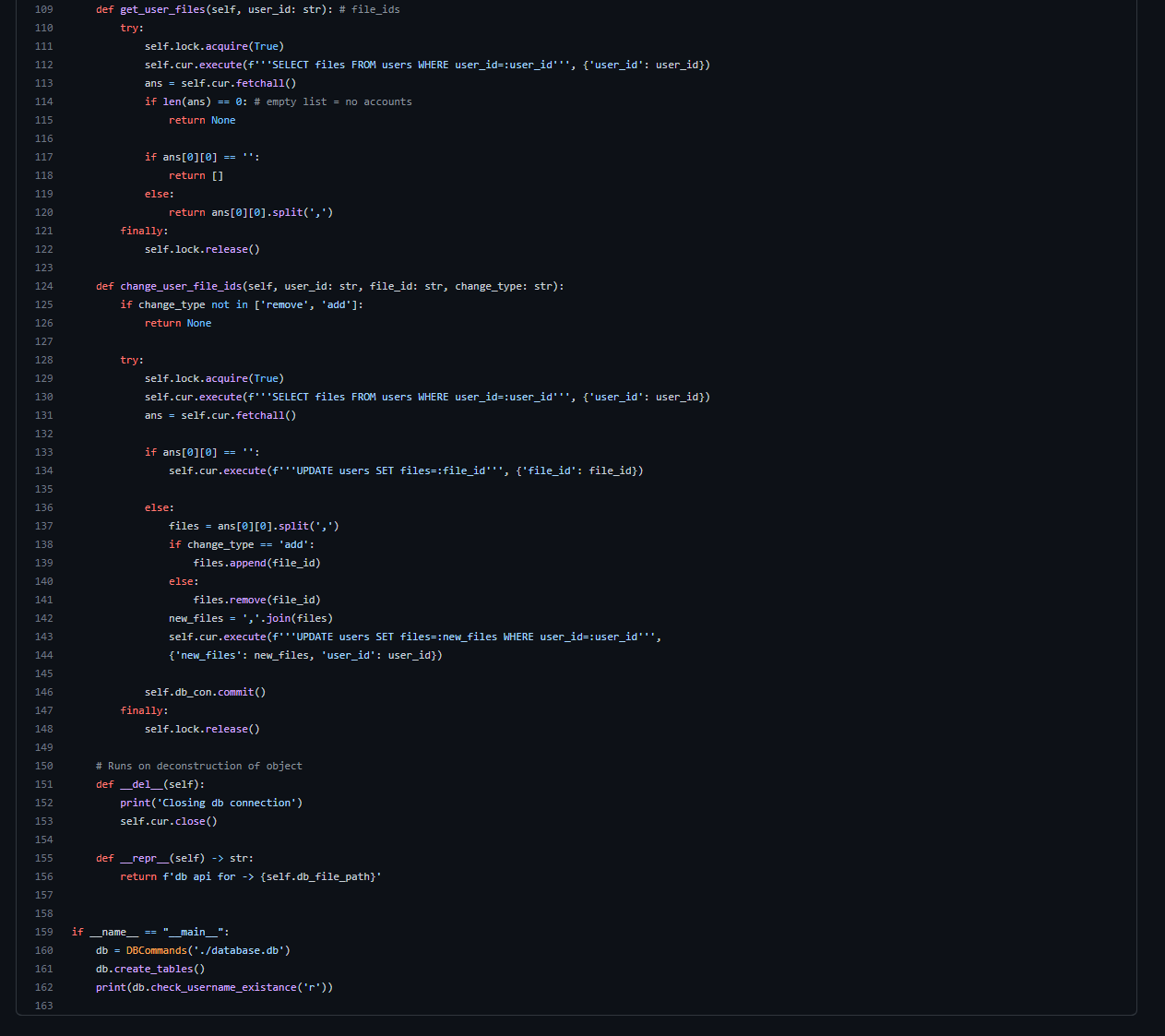
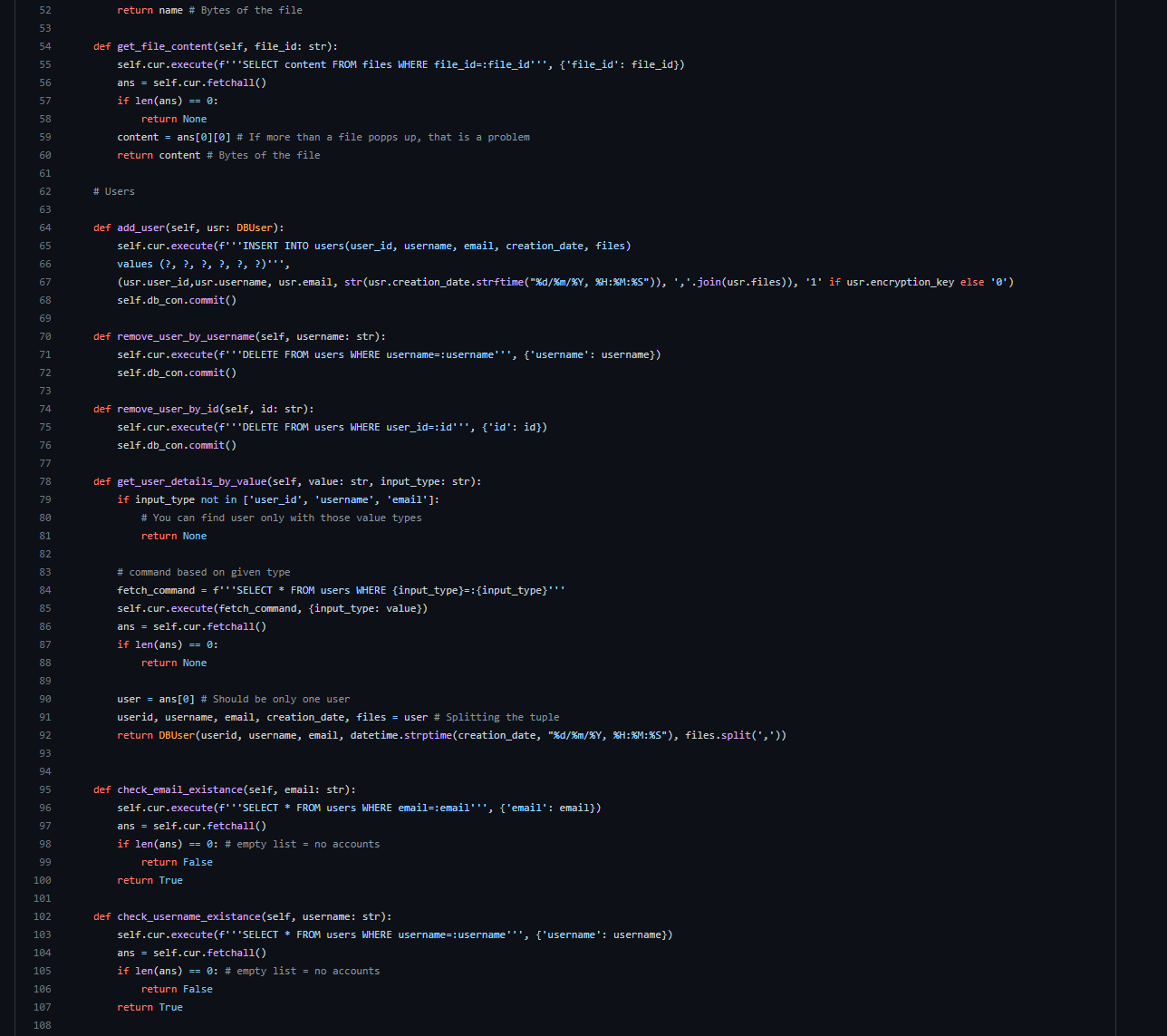
**מדריך למפתח**

תחת כותרת זו אסביר ואראה את המוח שמאחורי הפרויקט (כמובן שאני מדבר על המחשב). אני אראה קטעי קוד, קבצים שונים ואסביר עליהם את הרקע – שאדם שירצה אולי להמשיך ולפתח את התוכנה, יוכל לעשות זאת עם ההסברים פה.

**קובץ מס' 1 – DBCommands.py**

מהשם ניתן להבין שהוא אחראי על כל האינטגרציה עם מבנה הנתונים. הקוד שמטפל במאגר הנתונים משתמש במנעול אחד לכל המאגר, שלא ייגשו אליו שני משתמשים בו זמנית, ויפגעו במידע. לא אפרט בקובץ זה אך [מנעול](https://he.wikipedia.org/wiki/%D7%9E%D7%A0%D7%A2%D7%95%D7%9C_(%D7%AA%D7%95%D7%9B%D7%A0%D7%94)) משומש בשביל לשמור על משאב משותף לכמה תהליכים שרצים בו זמנית. לדוגמה, אם משתמש א' ומשתמש ב' יוספו למערכת בקירוב באותו הזמן, המערכת לא תדע את מי מהם לשמור. לכן, אחד מהמשתמשים ינעל מנעול על מאגר הנתונים, ולאחר שיסיים להשתמש בו – ישחרר את המנעול וייתן למשתמש אחר לגשת.





**קובץ מס' 2 – DBObjects.py**

בד"כ כשיוצרים מאגר נתונים, יוצרים אובייקטים שמטרתם לייצג את הטבלאות במאגר. כלומר אני יצרתי אובייקט למשתמש ואובייקט לקובץ, בשביל שאוכל לעבוד עם הקבצים שאני מכניס למאגר, בצורה תכנותית.



**רפלקציה**

**ביבליוגרפיה**

**נספחים**