

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 המשותמשות בעולם. כל כך משומשת, עד שהיא שינתה את הדרך שבה אנו מעבירים קבצים ממחשב למחשב. אמנם גוגל דרייב מצוין 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 בPromises, 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 (מהמחזורת 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...

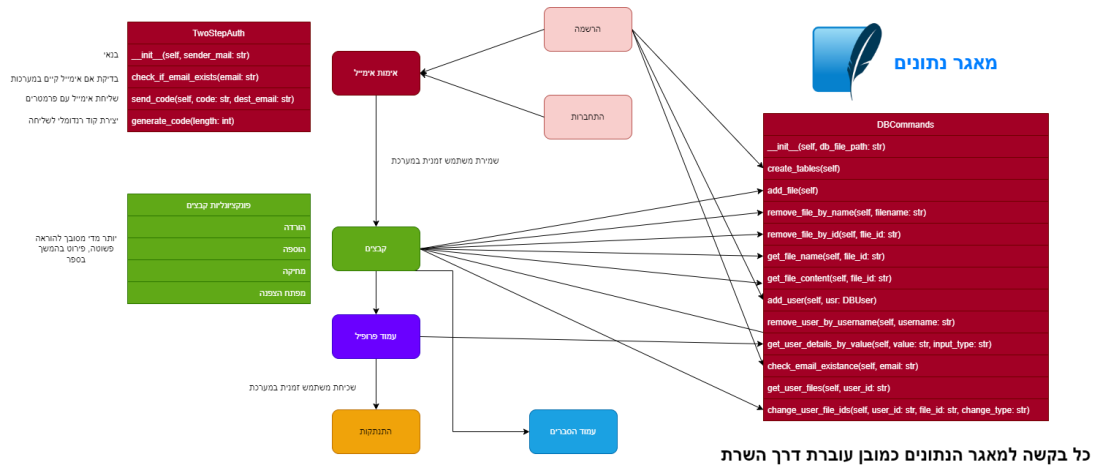
פיתוחים עתידיים

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.

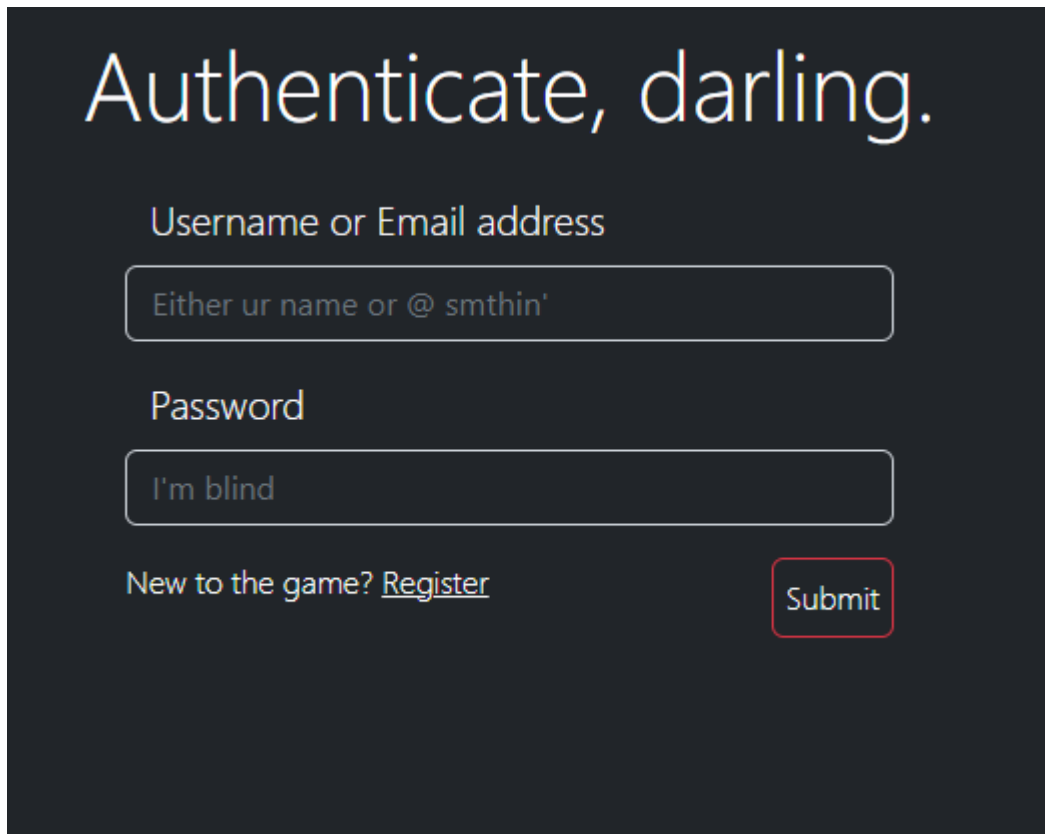
מדריך למשתמש

התקנת התוכנה

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

הרצת התוכנה

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



Authenticate, darling.

Username or Email address

Either ur name or @ smthin'

Password

I'm blind

New to the game? [Register](#)

Submit

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

Authenticate, darling.

Email address

Username

Password

Have an account? [Login](#)

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

Authenticate, darling.

Email address

Username

Password

ilan147963@gmail.com already exists!

Have an account? [Login](#)

Authenticate, darling.

Username or Email address

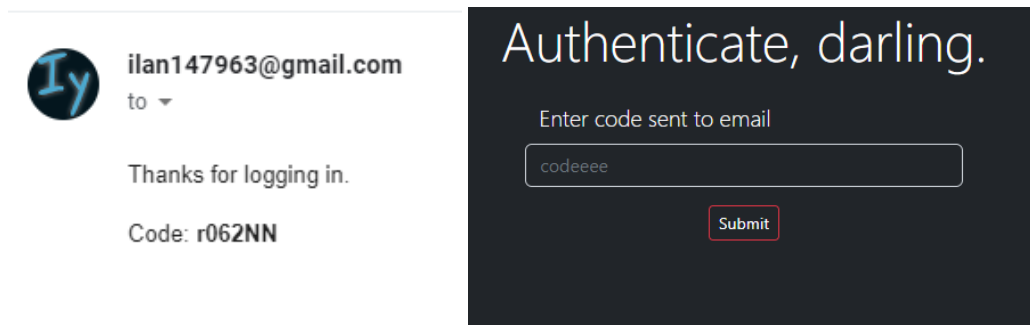
Password

One of the credentials is incorrect.

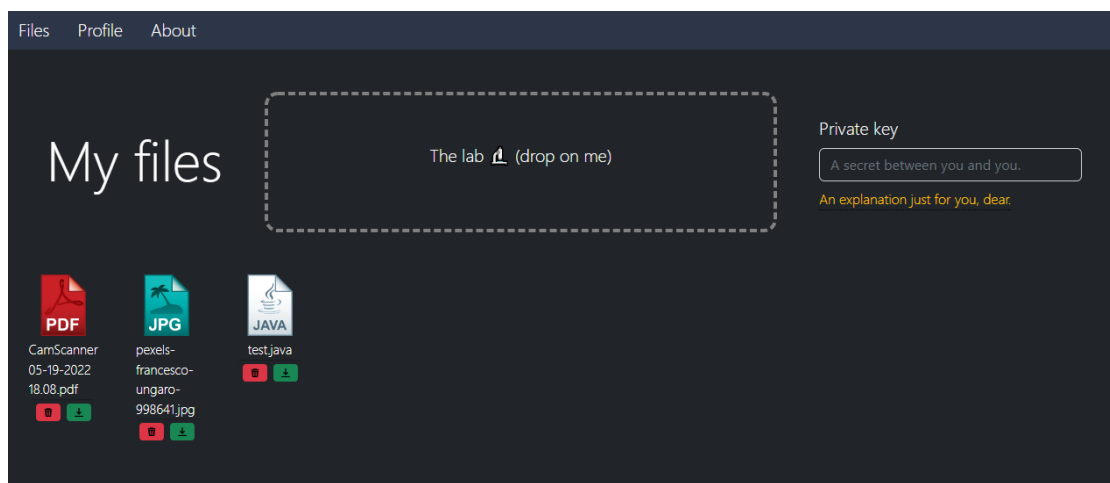
New to the game? [Register](#)

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').

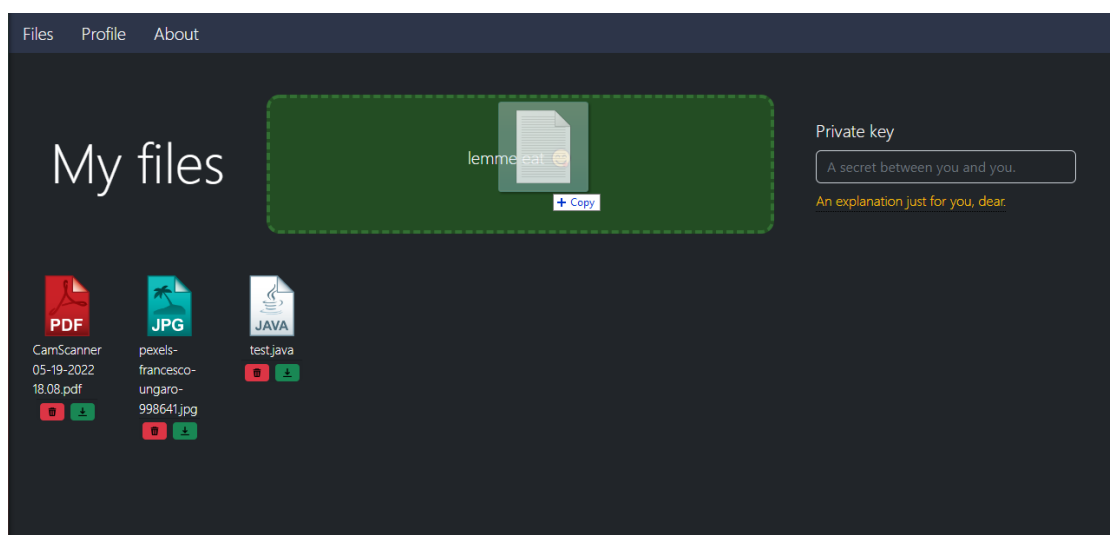
SafeCloud



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

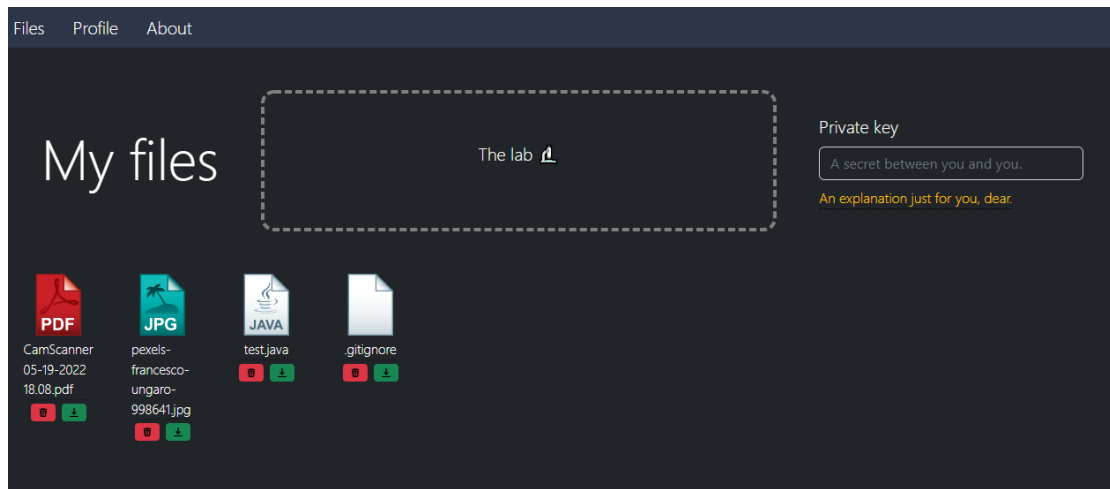


נכנסתי למשתמש שכבר יש בו קבצים, אז זה טען את הקבצים הקודמים. אם נרצה להוסיף קובץ, נגרור את הקובץ לתוך המסגרת האמצעית:

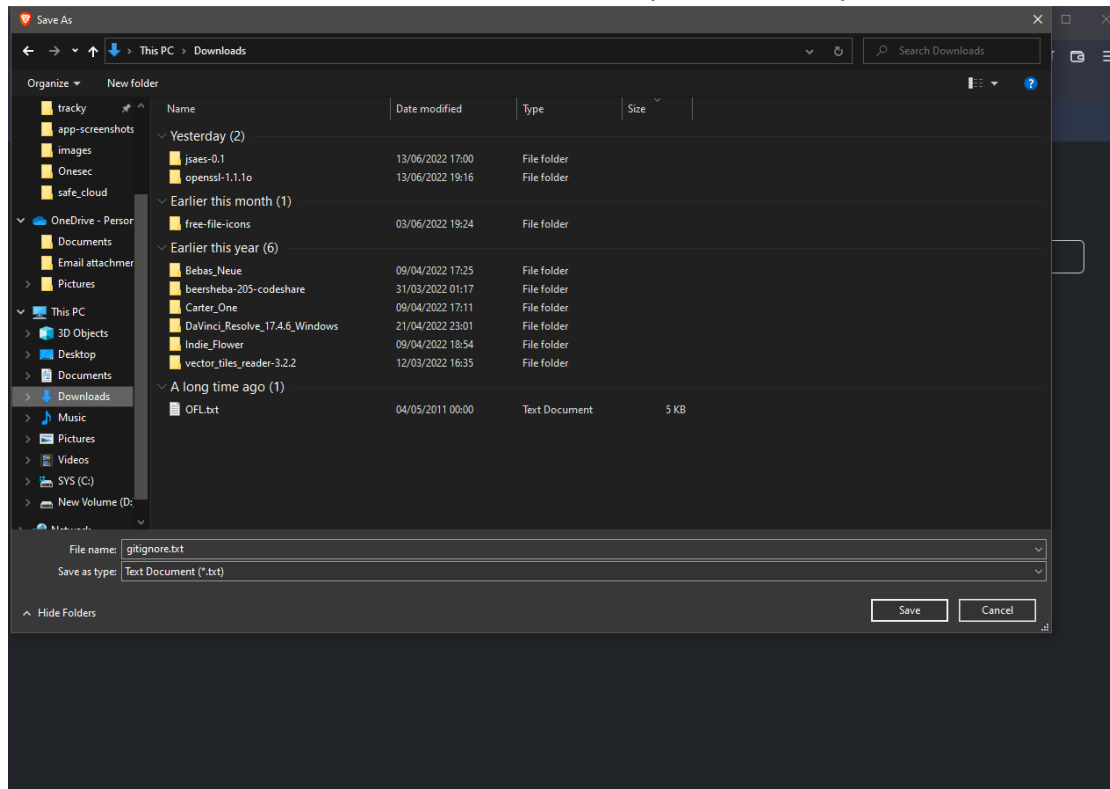


נראה שהקובץ אכן נוסף:

SafeCloud

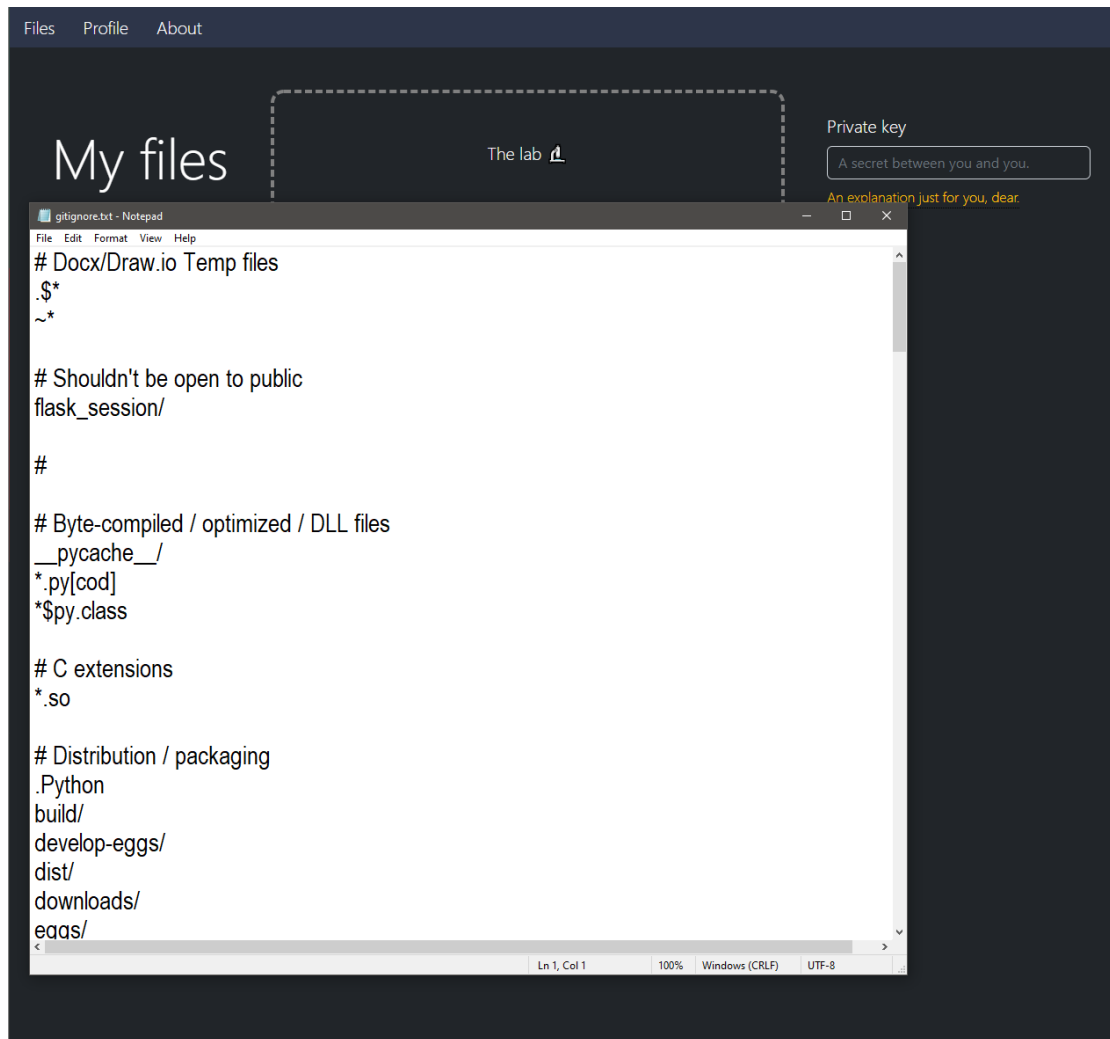


בנוסף, אם נרצה להוריד אותו, נלחץ על הלחצן הירוק ולמחיקה נלחץ על הלחצן האדום.
בעת הורדה האתר יבקש ממנו את מיקום ההורדה:

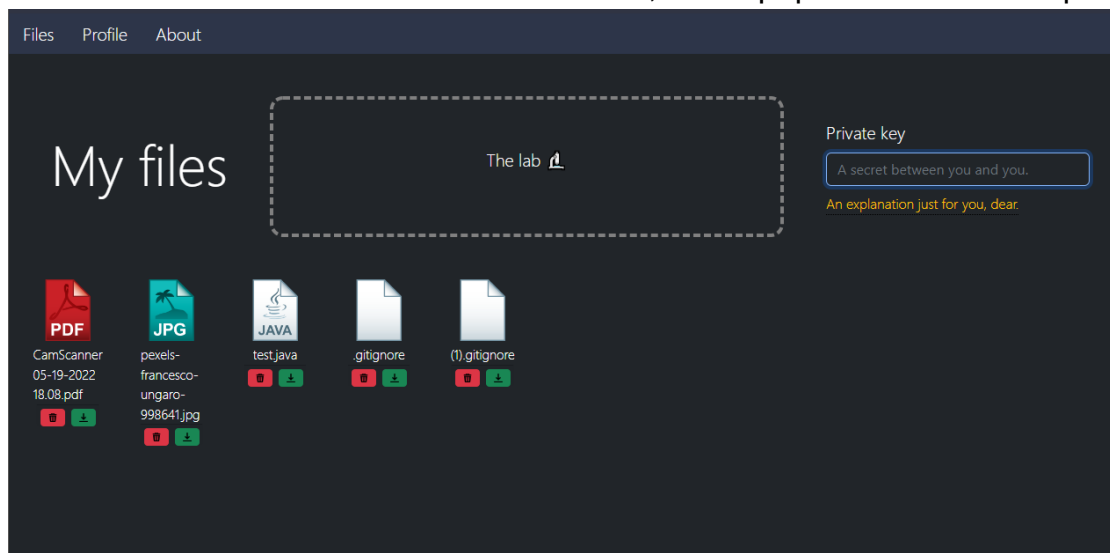


ונראה שהקובץ אכן הורד כמו שצריך.

SafeCloud

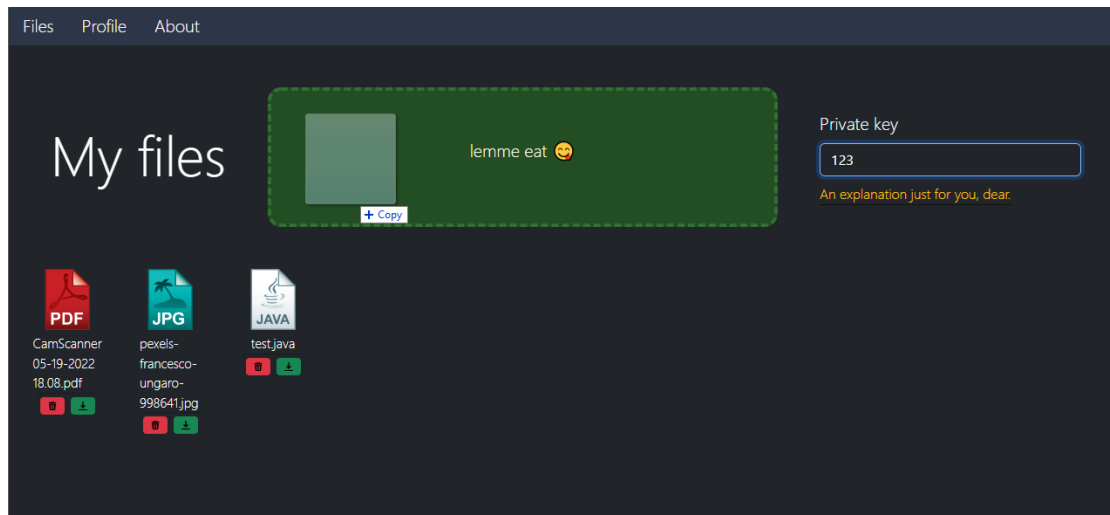


במקרה שנכניס את אותו קובץ פעמיים, נראה שהשם משתנה בהתאם:

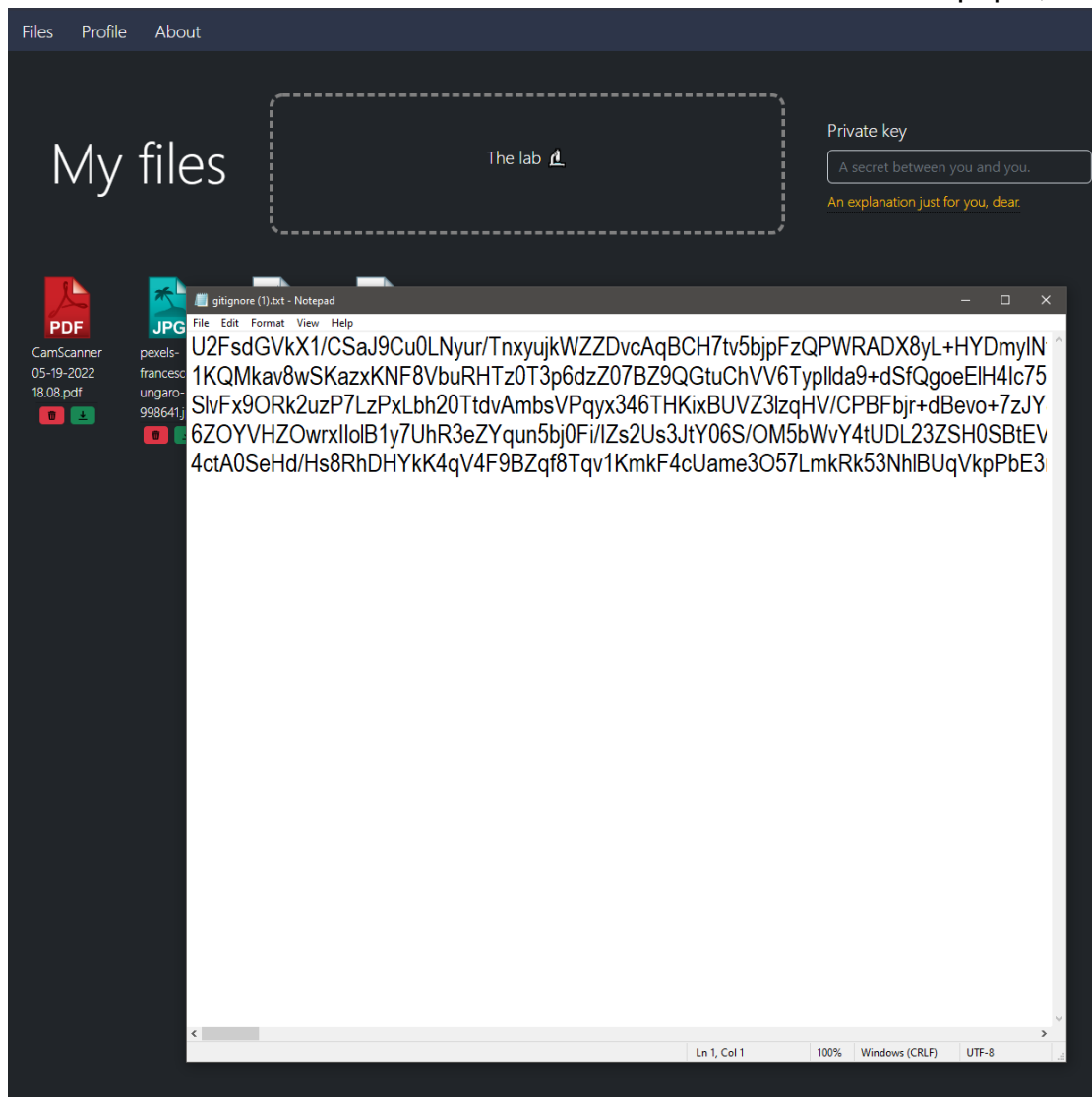


כמו כן, לדעתי הפיצ'ר הכי חזק בתוכנה הזאת, שאנחנו יכולים להצפין קובץ מצד הלקוח. כלומר, להצפין את הקובץ עוד לפני שיוצא מהמחשב שלנו – וכך אף אחד מלבדנו לא יוכל לדעת מה יש בקובץ, אלא אם כן הוא **מחזיק בקוד שלנו**. יש לזכור קוד הצפנה ולהוסיף קובץ כשהוא מוקלד:

SafeCloud



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:

SafeCloud

The screenshot shows the SafeCloud web interface. At the top, there are tabs for 'Files', 'Profile', and 'About'. The main heading is 'My files'. Below it, there's a dashed box labeled 'The lab' with a link icon. To the right, there's a 'Private key' section with a text input field containing '123' and a link 'An explanation just for you, dear.'.

In the foreground, a Notepad window titled 'gitignore (2).txt - Notepad' is open. It contains the following text:

```
# Docx/Draw.io Temp files
.*$
~.*

# Shouldn't be open to public
flask_session/

#

# Byte-compiled / optimized / DLL files
__pycache__/*
*.py[cod]
*$py.class

# C extensions
*.so

# Distribution / packaging
.Python
build/
develop-eggs/
dist/
downloads/
eggs/
```

בום!

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

The screenshot shows the SafeCloud web interface. At the top, there are tabs for 'Files', 'Profile', and 'About'. The main heading is 'My files'. Below it, there's a dashed box labeled 'The lab' with a link icon. To the right, there's a 'Private key' section with a text input field containing 'A secret between you and you.' and a link 'An explanation just for you, dear.'.

In the foreground, a Notepad window titled 'gitignore (2).txt - Notepad' is open. It contains the following text:

```
# Docx/Draw.io Temp files
.*$
~.*

# Shouldn't be open to public
flask_session/

#

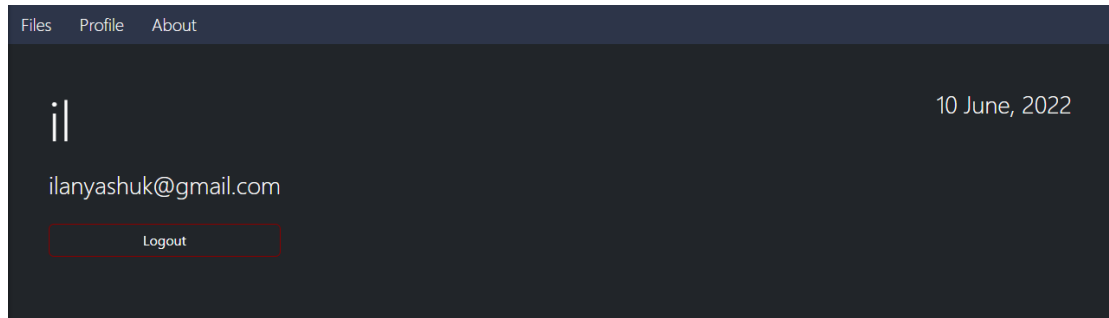
# Byte-compiled / optimized / DLL files
__pycache__/*
*.py[cod]
*$py.class

# C extensions
*.so

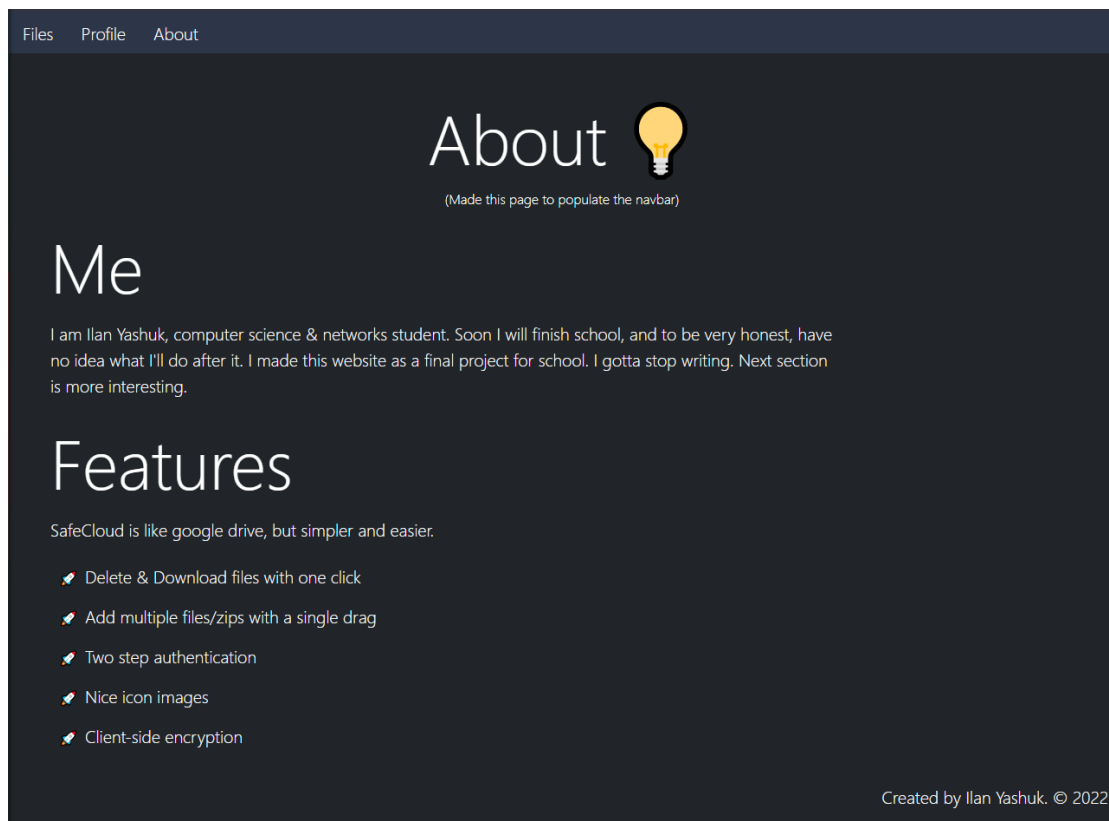
# Distribution / packaging
.Python
build/
develop-eggs/
dist/
downloads/
eggs/
```

SafeCloud

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.

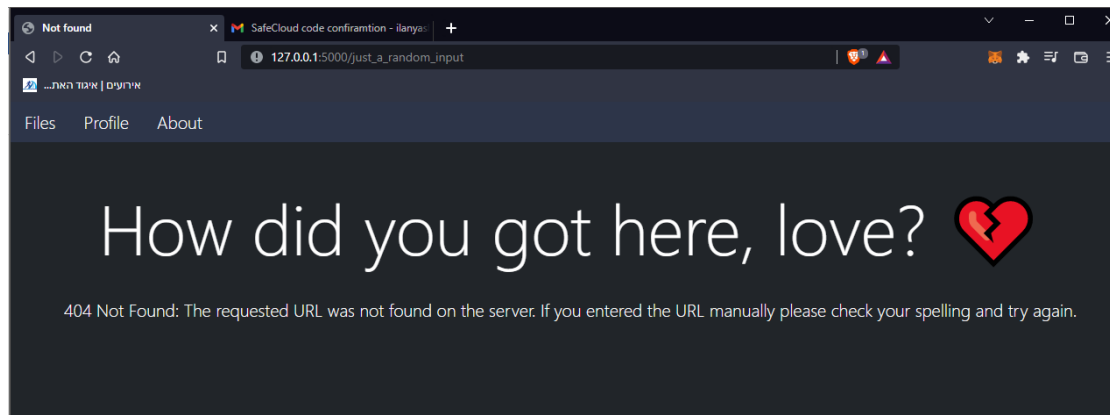


שני הדפים האחרונים הם דף שמסביר על פונקציונליות האתר:



ודף שגיאה:

SafeCloud



בסיס הנתונים – sqlite3

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

טבלת משתמשים (דוגמה)

Reset Filters Records: 2					
	user_id	username	email	creation_date	files
1	508179a3e96c081fcc...	il	ilanyashuk@gmail.com	10/06/2022, 00:13:48	d9efd378-e839-11ec...
2	18b02c6aa045a73e4...	ilan	ilan147963@gmail.co...	10/06/2022, 00:16:24	6e74353a-e839-11ec...

- 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)

Reset Filters Records: 27					
	file_id	in_dir	name	content	added_date
1	e5ab8748-e76c-11ec...		bin.png	10.56 KB	08/06/2022, 23:52:23
2	ee61ef3b-e76c-11ec...		mech.doc	137.5 KB	08/06/2022, 23:52:38
3	f094b5c8-e76c-11ec...		download.jpg	2.31 KB	08/06/2022, 23:52:42
4	4983daa6-e76d-11ec...		download.jpg	2.31 KB	08/06/2022, 23:55:11
5	4c35bfb5-e76d-11ec...		Document (1).docx	13.81 KB	08/06/2022, 23:55:15
6	4db8ff11-e76d-11ec...		Document (2).docx	13.81 KB	08/06/2022, 23:55:18
7	4f8e2d2d-e76d-11ec...		Document (1) (9).docx	13.81 KB	08/06/2022, 23:55:21
8	5a8ee64a-e76d-11ec...		download (11).jpg	2.31 KB	08/06/2022, 23:55:39
9	7158a37c-e76d-11ec...		download (13).jpg	2.31 KB	08/06/2022, 23:56:18
10	a97b4918-e76d-11ec...		-.pdf	212.59 KB	08/06/2022, 23:57:52
11	aaf95ec4-e76d-11ec...		-(17).pdf	212.59 KB	08/06/2022, 23:57:54
12	b85f9949-e76d-11ec...		-(18).pdf	212.59 KB	08/06/2022, 23:58:17
13	d98d60e3-e76d-11ec...		-(3).pdf	212.59 KB	08/06/2022, 23:59:12
14	ae821225-e776-11ec...		test.java	2.35 KB	09/06/2022, 01:02:26
15	5d1e0958-e777-11ec...		WhatsApp Image 20...	307.42 KB	09/06/2022, 01:07:19
16	2d749781-e778-11ec...		2.pdf	670.8 KB	09/06/2022, 01:13:08
17	c40166a9-e779-11ec...		database.db	3.75 MB	09/06/2022, 01:24:30
18	20f71ac3-e82f-11ec...		_blank.png	558 Bytes	09/06/2022, 23:02:45
19	22f46527-e82f-11ec...		file_icon.png	4.9 KB	09/06/2022, 23:02:48
20	2579cf2c-e82f-11ec...		Untitled document.d...	6.66 KB	09/06/2022, 23:02:53
21	15002038-e839-11ec...		Untitled document.d...	6.66 KB	10/06/2022, 00:14:00
22	16d541d5-e839-11ec...		white-stars-transpare...	43.46 KB	10/06/2022, 00:14:03

- 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). הנה הקוד:

```
16 def hashify_user(email: str, password: str, creation_date: datetime):
17     # Some dynamic salting for md5 hash
18
19     creation_time_mixed = str(creation_date.timestamp())[:-2]
20     salt = hashlib.md5(email.encode()).digest()[:2] + \
21     hashlib.md5(creation_time_mixed.encode()).digest()[:3]
22     encstr = hashlib.md5(password.encode() + salt).hexdigest()
23     return encstr
```

- קבלת אימייל, סיסמה, תאריך יצירה כקלט
- הפיכת זמן יצירה לכמות שניות מאז 1970 ובלבול המחרוזת
- יצירת מלח, 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](#) משמש 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.

```

163 lines (129 sloc) 6.25 KB
Raw Blame

1 from datetime import datetime
2 import sqlite3
3 import threading
4 from DBObjects import DBFile, DBUser
5
6 class DBCommands:
7     def __init__(self, db_file_path: str):
8         self.db_file_path = db_file_path
9         self.db_con = sqlite3.connect(db_file_path, check_same_thread=False)
10        self.cur = self.db_con.cursor() # Will be used to operate in db
11        self.lock = threading.Lock()
12
13    def create_tables(self):
14        self.cur.execute('CREATE TABLE IF NOT EXISTS files
15        (file_id text primary key, in_dir integer, name text, content blob, added_date text)')
16
17        self.cur.execute('CREATE TABLE IF NOT EXISTS users
18        (user_id text primary key, username text, email text, creation_date text, files text, encryption_key text)')
19
20        self.cur.execute('CREATE TABLE IF NOT EXISTS dirs
21        (dir_id text primary key, name text, files text)')
22
23        self.db_con.commit()
24
25    # Files
26
27    def add_file(self, file: DBFile):
28        try:
29            self.lock.acquire(True)
30            self.cur.execute(f'INSERT INTO files(file_id, in_dir, name, content, added_date)
31            values (?, ?, ?, ?, ?)',
32            (file.file_id, '1' if file.in_dir else '0', file.name, file.content, str(file.added_date.strftime("%d/%m/%Y, %H:%M:%S"))))
33            self.db_con.commit()
34        finally:
35            self.lock.release()
36
37    def remove_file_by_name(self, filename: str):
38        self.cur.execute(f'DELETE FROM files WHERE name=:filename', {'filename': filename})
39        self.db_con.commit()
40
41    def remove_file_by_id(self, file_id: str):
42        self.cur.execute(f'DELETE FROM files WHERE file_id=:file_id', {'file_id': file_id})
43        self.db_con.commit()
44
45    def get_file_name(self, file_id: str):
46        self.cur.execute(f'SELECT name FROM files WHERE file_id=:file_id', {'file_id': file_id})
47        ans = self.cur.fetchall()
48        if len(ans) == 0:
49            return None
50
51        name = ans[0][0] # If more than a file pops up, that is a problem
52        return name # Bytes of the file

```

SafeCloud

```
52         return name # Bytes of the file
53
54     def get_file_content(self, file_id: str):
55         self.cur.execute(f'''SELECT content FROM files WHERE file_id={file_id}''', {'file_id': file_id})
56         ans = self.cur.fetchall()
57         if len(ans) == 0:
58             return None
59         content = ans[0][0] # If more than a file pops up, that is a problem
60         return content # Bytes of the file
61
62     # Users
63
64     def add_user(self, usr: DBUser):
65         self.cur.execute(f'''INSERT INTO users(user_id, username, email, creation_date, files)
66         values (?, ?, ?, ?, ?)''',
67         (usr.user_id, usr.username, usr.email, str(usr.creation_date.strftime("%d/%m/%Y, %H:%M:%S")), ','.join(usr.files)), '1' if usr.encryption_key else '0')
68         self.db_con.commit()
69
70     def remove_user_by_username(self, username: str):
71         self.cur.execute(f'''DELETE FROM users WHERE username={username}''', {'username': username})
72         self.db_con.commit()
73
74     def remove_user_by_id(self, id: str):
75         self.cur.execute(f'''DELETE FROM users WHERE user_id={id}''', {'id': id})
76         self.db_con.commit()
77
78     def get_user_details_by_value(self, value: str, input_type: str):
79         if input_type not in ['user_id', 'username', 'email']:
80             # You can find user only with those value types
81             return None
82
83         # command based on given type
84         fetch_command = f'''SELECT * FROM users WHERE {input_type}={input_type}'''
85         self.cur.execute(fetch_command, {input_type: value})
86         ans = self.cur.fetchall()
87         if len(ans) == 0:
88             return None
89
90         user = ans[0] # Should be only one user
91         user_id, username, email, creation_date, files = user # Splitting the tuple
92         return DBUser(user_id, username, email, datetime.strptime(creation_date, "%d/%m/%Y, %H:%M:%S"), files.split(','))
93
94
95     def check_email_existance(self, email: str):
96         self.cur.execute(f'''SELECT * FROM users WHERE email={email}''', {'email': email})
97         ans = self.cur.fetchall()
98         if len(ans) == 0: # empty list = no accounts
99             return False
100         return True
101
102     def check_username_existance(self, username: str):
103         self.cur.execute(f'''SELECT * FROM users WHERE username={username}''', {'username': username})
104         ans = self.cur.fetchall()
105         if len(ans) == 0: # empty list = no accounts
106             return False
107         return True
108
109     def get_user_files(self, user_id: str): # file_ids
110         try:
111             self.lock.acquire(True)
112             self.cur.execute(f'''SELECT files FROM users WHERE user_id={user_id}''', {'user_id': user_id})
113             ans = self.cur.fetchall()
114             if len(ans) == 0: # empty list = no accounts
115                 return None
116
117             if ans[0][0] == '':
118                 return []
119             else:
120                 return ans[0][0].split(',')
121         finally:
122             self.lock.release()
123
124     def change_user_file_ids(self, user_id: str, file_id: str, change_type: str):
125         if change_type not in ['remove', 'add']:
126             return None
127
128         try:
129             self.lock.acquire(True)
130             self.cur.execute(f'''SELECT files FROM users WHERE user_id={user_id}''', {'user_id': user_id})
131             ans = self.cur.fetchall()
132
133             if ans[0][0] == '':
134                 self.cur.execute(f'''UPDATE users SET files={file_id}''', {'file_id': file_id})
135
136             else:
137                 files = ans[0][0].split(',')
138                 if change_type == 'add':
139                     files.append(file_id)
140                 else:
141                     files.remove(file_id)
142                 new_files = ','.join(files)
143                 self.cur.execute(f'''UPDATE users SET files={new_files} WHERE user_id={user_id}''',
144                 {'new_files': new_files, 'user_id': user_id})
145
146             self.db_con.commit()
147         finally:
148             self.lock.release()
149
150     # Runs on deconstruction of object
151     def __del__(self):
152         print('Closing db connection')
153         self.cur.close()
154
155     def __repr__(self) -> str:
156         return f'db api for -> {self.db_file_path}'
157
158
159 if __name__ == "__main__":
160     db = DBCommands('./database.db')
161     db.create_tables()
162     print(db.check_username_existance('r'))
163
```

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.

```
24 lines (20 sloc) | 834 Bytes

1  import datetime
2
3  class DBFile:
4      def __init__(self, file_id: str, in_dir: bool, name: str, content: bytes, added_date: datetime.datetime):
5          self.file_id = file_id
6          self.in_dir = in_dir
7          self.name = name
8          self.content = content
9          self.added_date = added_date
10
11     def __repr__(self):
12         return f'{self.file_id} -> {self.name}, {self.added_date}'
13
14     class DBUser:
15         def __init__(self, user_id: str, username: str, email: str, creation_date: datetime.datetime, files: list[str], encryption_key=''):
16             self.user_id = user_id
17             self.username = username
18             self.email = email
19             self.creation_date = creation_date
20             self.files = files
21             self.encryption_key = encryption_key
22
23         def __repr__(self):
24             return f'{self.user_id} -> {self.username} ({self.email})'
```

רפלקציה

ביבליוגרפיה

