DATAWALLET

שם התלמיד: שי קרן
ת.ז: 215306283
בית ספר: עמל חדרה

תוכן עניינים

מבוא

הפרוייקט הזה נועד לרכז מידע רפואי בשרת מאובטח. כלומר, לקחת מידע רפואי של מטופל מכמה מוסדות שעוסקים ברפואה ולרכז אותו בשרת אחד באופן מאובטח. כמו כן, רופא יכול להגיש בקשה, שאותה המטופל יכול לאשר. וכך למטופל ולרופאים שלו יש גישה לתיק הרפואי המלא שלו, מכל הבתי חולים והמרפאות בהם טופל. באופן כזה רופא יכול לקבל תמונת מצב מלאה ולהחליט על טיפולים המתאימים ביותר למצב.

המטרה העיקרית של הפרוייקט הייתה להצליח לעשות את זה באופן כזה שהשרת עצמו מחזיק רק את הקבצים המוצפנים, כלומר, גם למי שיש גישה מלאה לשרת, לא יכול לקרוא את המידע הרפואי שנמצא בו.

בהתחלה נתקלנו במספר בעיות שנאלצנו לפתור באופן תאורטי, עוד לפני ההתחלה של התכנות של הפרוייקט. כמו השיחזור סיסמא, הרי השרת לא מחזיק את הסיסמא של המטופל שפותחת את המפתח הצפנה או את המפתח הצפנה עצמו. לאחר שמצאנו פיתרון החלטנו שהפרוייקט מסובך וכלל הרבה עבודות עם הצפנה ותקשורת, ובפייתון יש הרבה ספריות מוכנות בשבילם ולכן בחרנו לתכנת בפייתון, בסביבת עבודה שכבר עבדנו בעבר בשם PyCharm. התחלנו לחשוב על ההצפנות המתאימות ולהבין איך הן עובדות (פרוטוקול RSA להחלפת מפתחות ו- TwoFish להצפנה עצמה) נתקלנו בבעיות איתם (זמן ריצה ב- RSA ותהליך הפיענוח ב- TwoFish) והחלפנו הצפנה (ECC להחלפת המתפחות ו- AES להצפנה עצמה). פירקנו את הפרוייקט ל- 3 חלקים. צד שרת, צד לקוח, ועוד שרת שמדמה את פעולותיו של השרתים של הבתי חולים. החלטנו שהתקשורת תהיה באמצעות סוקטים בפרוטוקול TCP. חוץ מזה, המידע הרפואי שיעבור יהיה מאובטח וכל הקבצים עוברים ברשת בצורה מוצפנת. בנינו את הפרוייקט מבפנים החוצה. כלומר, במקום לבנות הכל בנפרד ולחבר הכל בסוף, בנינו גרסאות בסיסיות של הכל ועליהם הוספנו בכל פעם את הפיצ'רים והאפשרויות החדשות. בהתחלה אני עבדתי בעיקר על החלק הטכני, בניה של הלרוח והשרתים והתקשורת בינהם. ושרגא עבד על למידת ההצפנות ומימוש של מצפין ומפענח באמצעותם ובנייה של דרך להחליף מפתחות. לאחר מכן הוא עבד על בנייה של ה- DB (SQLite שכבר הכרנו והתאים לצרכים שלנו). ובהמשך השתלב בבניה של השרתים והאפשרויות של הלקוח.

כל בית חולים מאחסן מידע באופן רפואי ושרתים שמאחסנים מידע לארגונים ספציפיים אבל לא באופן יעיל במיוחד. בנוסף, יש הרבה שרתים לשמירת מסמכים משלל סוגים – מנפוצים ובסיסים כמו גוגל דרייב ועד כאלה מאובטחים יותר כמו Microsoft ignite שדי דומה ברעיון שלו לשלנו.

החידוש שבפרוייקט שלנו הוא שאנחנו, בתור בעלי גישה מלאה לשרת ולכל המידע השמור בו, לא יכולים לקרוא כלום וכמות המידע שאנחנו יכולים לגשת אליו כמעט לא קיימת, בעיקר שמות של המסמכים וכד'. כלומר, אפילו אם מישהו פורץ לשרת שלנו. המידע שהוא יכול לגשת אליו הוא לא חשוב במיוחד.

כמו שתיארתי, הבעיה העיקרית של הפרוייקט הייתה היכולת לאפשר רק ללקוח גישה למידע המלא, ולא למי שאחראי \ הבעלים של השרת. ולמרות זאת, עדיין לאפשר ללקוח דברים כמו שיחזור סיסמא ושיחזור מידע.

ישנם שתי סיבות שבחרתי בפרוייקט הזה: הארשונה היא שבמגשימים, למרות הידע הרב שכן קיבלתי, בקושי נגענו בתחום של אבטחת מידע, הפרוייקט הזה מתעסק בהצפנות ואבטחת מידע ולי היה חשוב להשלים וללמוד את הנושא הזה. הסיבה השנייה היא אישית, הייתי פעם בביקור בבית חולים, הרופא שהיה שם ביקש מידע ישן מבית חולים אחר, ולבסוף אחרי השקעה של כסף, זמן ומאמץ קיבלנו רק מידע חלקי וזה היה ממש מתסכל. אם כל המידע הרפואי עלי היה מתרכז לשרת יחיד, כל מה שהייתי צריכה זה לאשר לרופא גישה אל זה, בכמות מזערית של השקעה.

הפיתרון לבעיה הוא שמירה של המפתחות של הלקוח בשרת הרפואי ולא לתת לשרת גישה אליהם בכלל. כל ההתעסקות עם ההצפנה נעשת רק בשרתים של המוסדרות הרפואיים ובקוד של הלקוח.

בנוסף, הפרוייקט מתסעסק במידע רפואי כי הוא אישי ופרטי, והסודיות שלו מוגנת בחוק (אנחנו יודעים שיש מסמך והנחיות בשם HIPPA שכוללים איך לאבטח מידע רפואי ומה הרמת אבטחה, אבל לא נגענו בזה כי אנחנו רק שתיים ויש חברות שלמות שמתעסקות ביצירת אבטחה למידע רפואי) כי זה הופך את הפרוייקט למוגדר יותר ונותן לנו את היכולת לעבוד עליו בצורה פחות כללית ויותר מכוונת מטרה.

ההאמת שזה פרוייקט די גמיש, המושג אבטחת מידע הוא מאוד כללי. למדנו על אבטחת מידע, התכנון היה להוסיף גם חלקים נגד בעיות באבטחה ברשת (לדוגמא הגנה בסיסית מפני התקפת DDos) אבל הפרוייקט הזה גם ככה ממש גדול ומסובך. ופשוט לא היה לנו את הזמן להתעסק מעבר לתכנון הראשוני שלנו. אם כבר, הורדנו כמה חלקים שתכננו לבנות. כגון האפליקציה.

בשביל הפרוייקט הזה הסתמכנו על ידע קודם שהיה לנו משפות תכנות אחרות וגם מפייתון, כגון: Thraeds, סוקטים ועבודה עם SQLite. וגם למדנו דברים חדשים, כמו מה זה הצפנה א-סימטרית ואיך היא פועלת. למדנו איך עובדת הצפנת ECC ועל הצפנת AES. למדתי איך לשלוח אימיילים דרך פייתון ולמדתי איך לתקשר עם שרת באופן לא ישיר – כמו למשל תקשורת של הלקוח עם השרת של הבית חולים דרך השרת שלנו. פרוקוטול התקשורת הוא TCP שהוא פרוטוקול אמין.

מסמך יזום

תיאור כללי:

לפרוייקט יש שלושה חלקים: צד לקוח – המטופל \ הרופא, הצד של השרת של הפרוייקט ועוד שרת נוסף שמדמה את השרת הרפואי. בצד לקוח המבצד לקוח המטופל יכול הירשם ולקבל את הקבצים שלן, הרופא יכול לשלוח בקשה ע"פ ת"ז מטופל והמטופל יכול לאשר את הבקשה. צד השרת שומר את המידע הרפואי מוצפן ומשמש צינור בין הצד לקוח לשרת הרפואי. השרת הרפואי יוצר מפתחות משותפים עם הלקוח, דואג להצפנה של המסמכים במפתח ושליחה שלהם לשרת.

מטרת הפרוייקט:

לרכז מידע רפואי משרתים של מוסדות רפואיים בצורה מוצפנת ולאפשר למטופלים ולרופאים שלהם גישה לכל המידע השלם שלהם, בלי קשר לאיזה מוסד הם מטופלים בו כיום.

הליבה הטכנולוגית:

היכולת לאחסן מידע רפואי במקום אחד באופן בטוח ולוודא שיוכלו לגשת אליו רק מי שיש להם הרשאה לכך. תקשורת בין שרת ללקוח והצפנת ECC.

טכנולוגיות עיקריות ושיקולים עיקריים:

על מנת לממש את הליבה אנו נצטרך להשתמש הצפנת מפתחות (פרוטוקול ECC).הפרוטוקול יאפשר לנו להעביר מסרים בין הבית חולים למטופל בלי שהשרת או כל אדם אחר שיאזין באמצע יוכל לייצר לעצמו מפתח פיענוח. בחרנו בו כדי למנוע מבעלי השרת יכולת לגשת למידע וכדי להעביר אותו באינטרנט באופן בטיחותי. אנחנו נשתמש גם בפרוטוקול TCP לתקשורת בין שרת ללקוח. כמו כן, נשתמש בהצפנת AES כדי להצפין את הקבצים במפתחות שיצרנו ע"י ECC. שתי ההצפנות נבחרו בזכות היעילות שלהן, הספריות שיש להם בפייתון וקלות היכולת להצפין ולפענח באמצעותן.

אתגרים טכנולוגיים ומקורות:

1. אחסון הסיסמא (או המפתח הפרטי) באופן כזה שלא נוכל לגשת אליו כדי שלא נוכל לקרוא את המסמכים המוצפנים אצלנו. ועדיין יהיה אפשר לשנות סיסמא במקרה הצורך אצל הלקוח. המפתח הצפנה יישמר אצל המשתמש באופן מוצפן ע"י הסיסמה. זאת אומרת שרק בעזרת הסיסמא אפשר לקרוא את המפתח הפרטי. נצטרך ללמוד להצפין בעזרת סיסמאות.
2. אחסון והעברת מידע בינארי. בכדי להעביר מידע כמו קבצי PDF נצטרך ללמוד להעביר מידע בינארי בהצפנה.

קהל היעד: הפרויקט נועד לעזור למטופלים רפואיים ולרופאים.

דרישות חומרה: אין (זה פרויקט תיכנותי בלבד ללא רכיבי חומרה.)

פתרונות קיימים:

כל בית חולים מאחסן מידע באופן רפואי ושרתים שמאחסנים מידע לארגונים ספציפיים אבל לא באופן יעיל במיוחד. בנוסף, יש הרבה שרתים לשמירת מסמכים משלל סוגים – מנפוצים ובסיסים כמו גוגל דרייב ועד כאלה מאובטחים יותר כמו Microsoft ignite שדי דומה ברעיון שלו לשלנו.

מסמך אפיון

פיצ'רים ותהליכים עיקריים:

1. קבלת מידע משרת אחר ואחסונו אצלנו - קבלת מידע מוצפן של לקוח משרתי ארגונים רפואיים ואחסונו במקום המתאים שמסווג לפי מטופל. לקוח הוא המטופל במערכת שלנו. יש לו מגוון תיקים רפואיים מפוזרים בשרתים של מוסדות רפואיים שונים. שאותם אנו מקבצים למסד נתונים אחד. אנו עושים זאת ע"י הפצה של בקשות לפי ת"ז והצפנה שלהם והעברה ללקוח.

2. כניסה מאובטחת של מטופל \ רופא - רופא הוא לא באותו מעמד של מטופל. רופא יכול לגשת למסמכים אישיים של מטופלים. לכל רופא יש גם חשבון בצורת מטופל. רופא מאופיין ב- DB בעזרת ת"ז, אבל יכול להתווסף ל- DB רק במקרה שיש לו תעודה רפואית.

3. אישורים וסיווגים להברות מידע בין מטופל לרופא - בתחילת כל טרנזקציה בין רופא למטופל יש אישור של המטופל לרופא לגשת למסמכים המתאימים.

4. פתיחת הצפנה בצד לקוח באמצעות מפתח - ללקוח יש את המפתח שלו שבו הוא משתמש בכדי לפתוח את ההצפנה. המפתח לא יועבר ברשת.

5. שכחתי סיסמא - החלטנו קרוב להתחלה שלשרת לא תהיה גישה למידע השמור בתוכו. כדי לשמור על כך, לשרת לא היה את הסיסמאות של המשתמשים. הבעיה הנוצרת מזה הוא: אם משתמש שכח סיסמא, אין לו דרך למצוא אותה בחזרה, כי לשרת אין אותה. המערכת שעליו חשבנו כדי לפתור את זה פועלת בזה שהמפתח שמור אצל הבית חולים. כשמשתמש שוכח סיסמא, הוא מבקש מההשרת (לאחר הכנסת קוד אימות שנשלח באימייל) ליצור חדשה, השרת בתורו מבקש מהבית חולים את המפתח כשהוא מוצפן ע"י מפתח אחר שהבית חולים יוצר עם הלקוח ושולח אותו ללקוח שאיבד את הסיסמא. עכשיו הלקוח מפענח את המפתח הישן באמצעות החדש ויוצר סיסמא חדשה.

טכנולוגיות:

פיצ'ר/תהליך	טכנולוגיות ושפות תכנות
הצפנה	ECC- library in python AES- library in python
אחסון	SQL database
תעבורה	Socket / TCP
הצגת קבצים שונים	--
דיבור למייל	SMTP server- library in python

מבנה בסיס נתונים

ישנם 3 בסיסי נתונים שבהם הסרבר שלנו משתמש. בסיס נתונים של המידע הרפואי, של המשתמשים המטופלים, ושל הרופאים.

מה שבולט הוא ה- ID של כל שורה במסד נתונים. שהוא המפתח הראשי.

מסד נתונים של המטופלים:

שם המטופל
אימייל
ת"ז
מספר טלפון
מספר עזר למקרה ששכחו סיסמא
מפתח הצפנה מוצפן ע"י סיסמא
ת.ז. של איש קשר למקרה חירום
מסר מוצפן בשביל הבדיקה של ההתחברות

מסד הנתונים של המסמכים:

ת"ז של משתמש אליו קשור המסמך
שם המסמך
המסמך עצמו

מסד הנתונים של הרופאים (הוא חלק מהשורה בטבלה של מטופל, אצל מטופל רגיל הם פשוט NULL):

שם הרופא
מקום עבודה
מספר תעודת רפואה
התמחויות
כל המפתחות של המטופלים שאישרו לו גישה

ארכיטקטורה

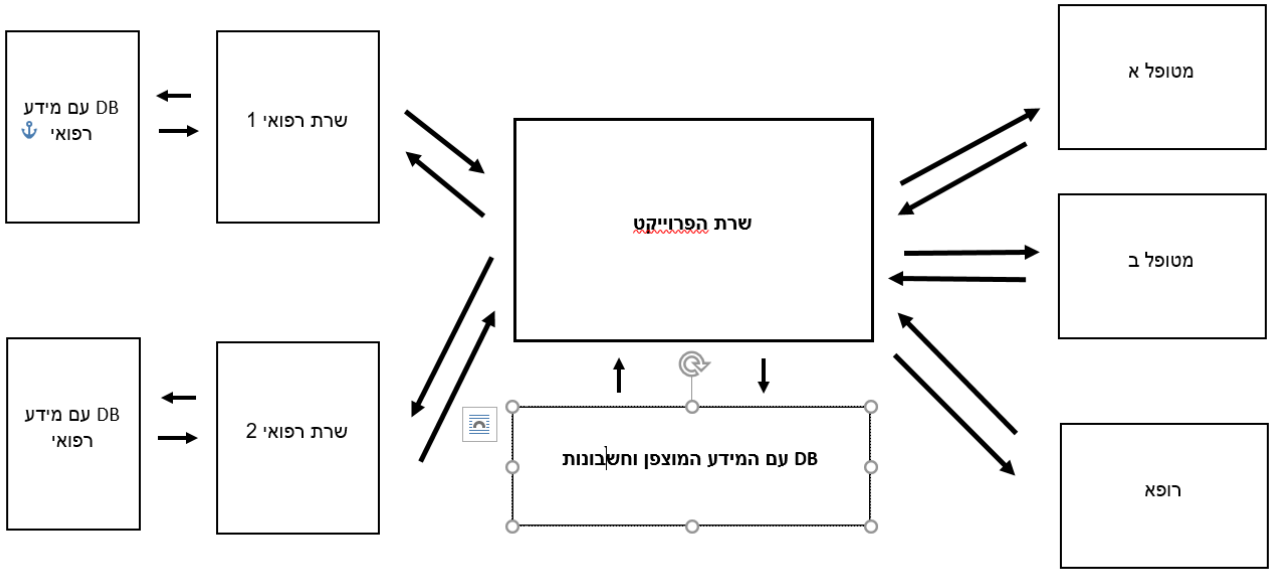
יש בפרויקט 3 חלקים עיקריים;

1. שרת - השרת שומר מידע מוצפן, ומספק אותו ללקוחות שצריכים אותו. הוא גם מנהל משתמשים, ללא גישה למידע האישי או לסיסמא שלהם.

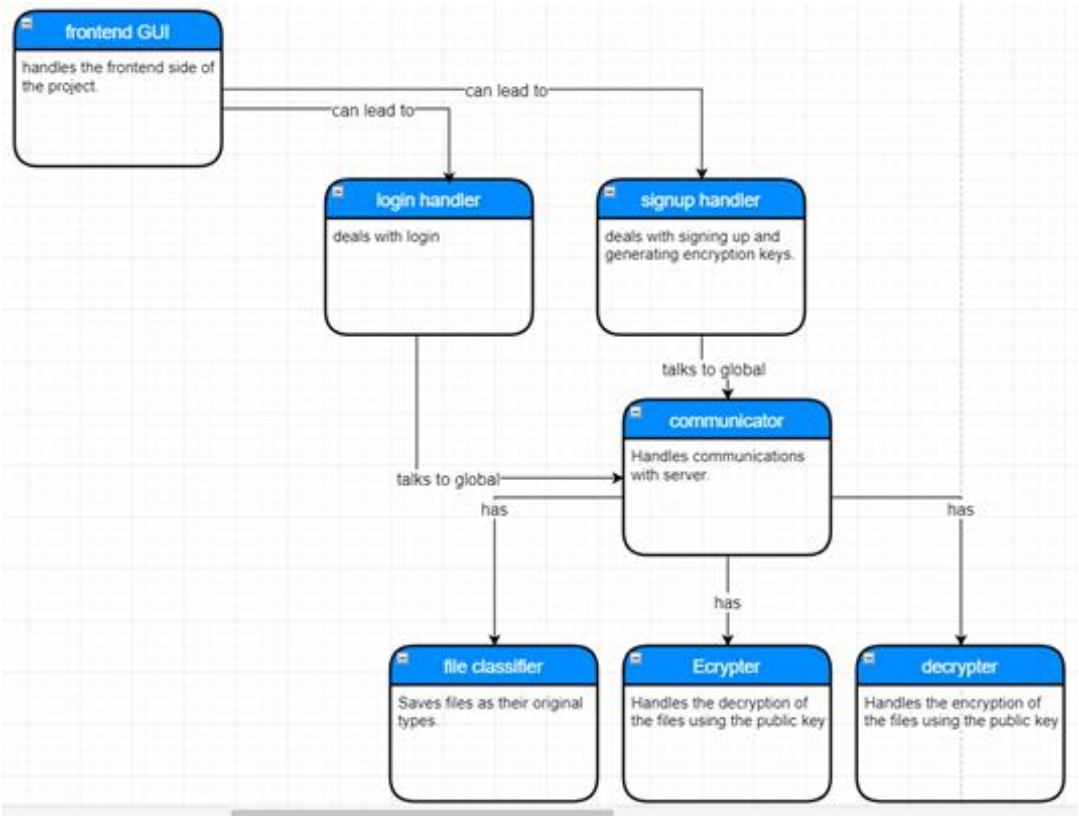
2. לקוח מטופל - ללקוח הזה משויכים מסמכים רפואיים. זה נעשה דרך הרצה של הפרוייקט ללא GUI בגלל מחסור בזמן.

3. לקוח רופא - הרופא צריך לקבל מידע רפואי ממטופלים, לבקש מידע, ולהוסיף למידע של מטופליו. לרופא אין גישה לאף מידע שלא אושר ע"י המטופלים שלו.

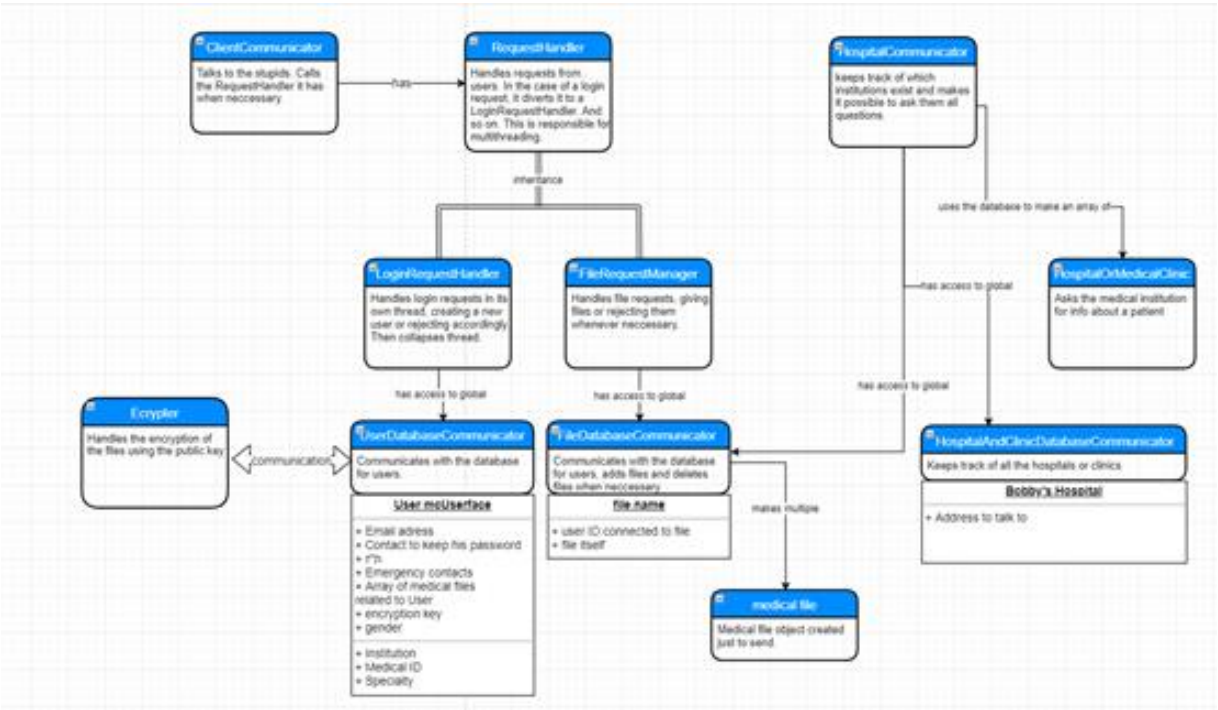
4. בנוסף, יש שרת דמה שאמור לדמות את הפעילות של שרת מידע רפואי. שהוא אמנם לא אמור להיות חלק גדול. אבל בגלל שנאלצנו לבנות אותו מאפס הוא חלק די גדול מהפרוייקט.



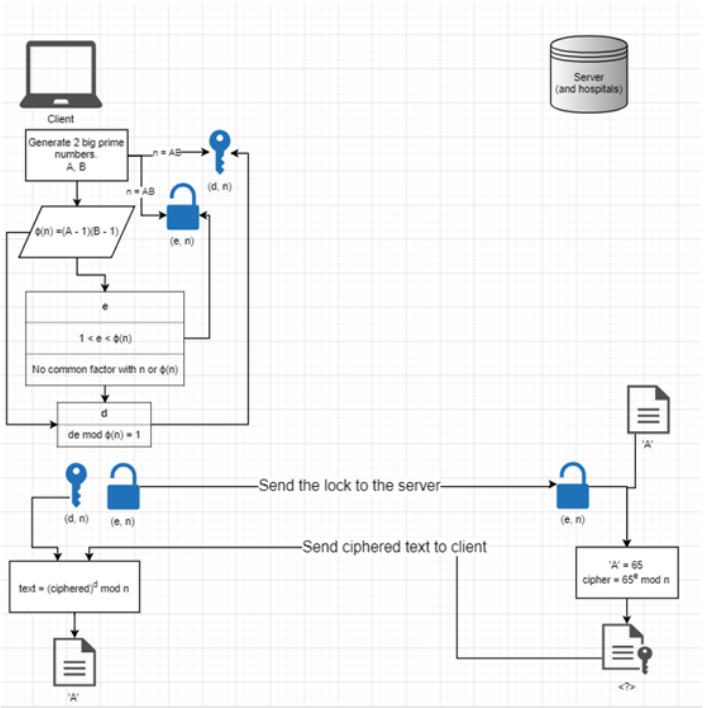
תיאור צד לקוח:



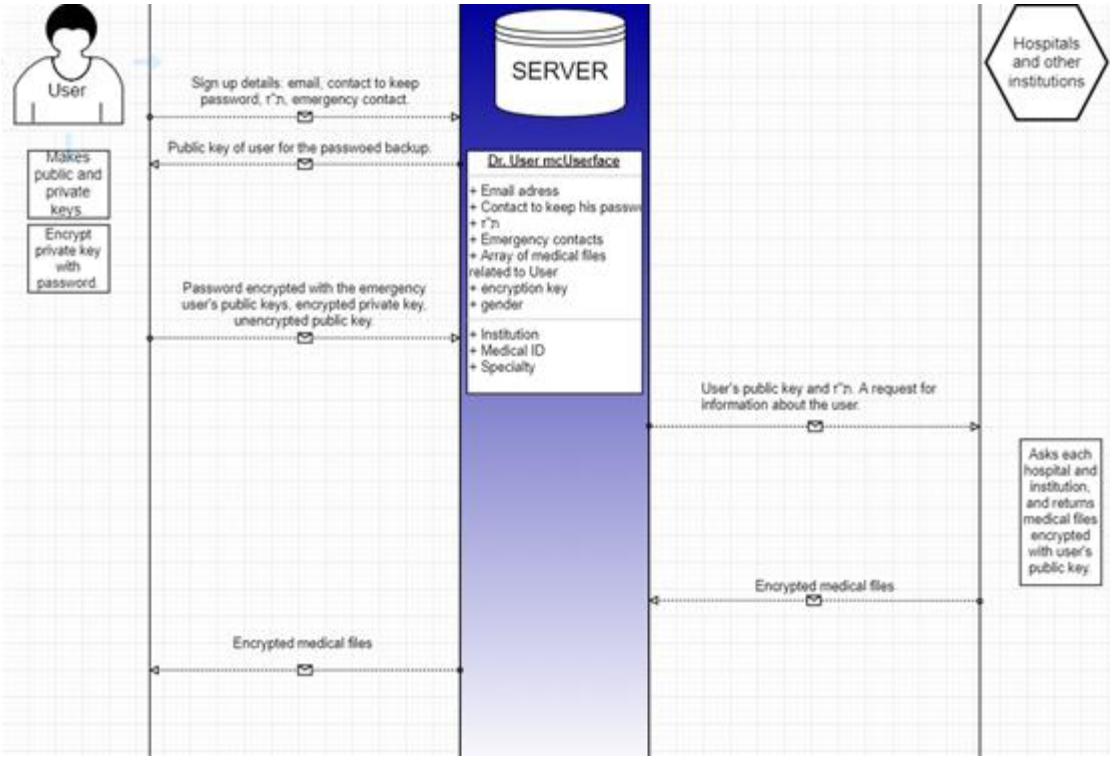
תיאור צד שרת:



תיאור הצפנת RSA:



תהליך ההרשמה:



תוכנית עבודה

מתן עדיפות לפיצ'רים, חלוקה לשלבים מסודרים:

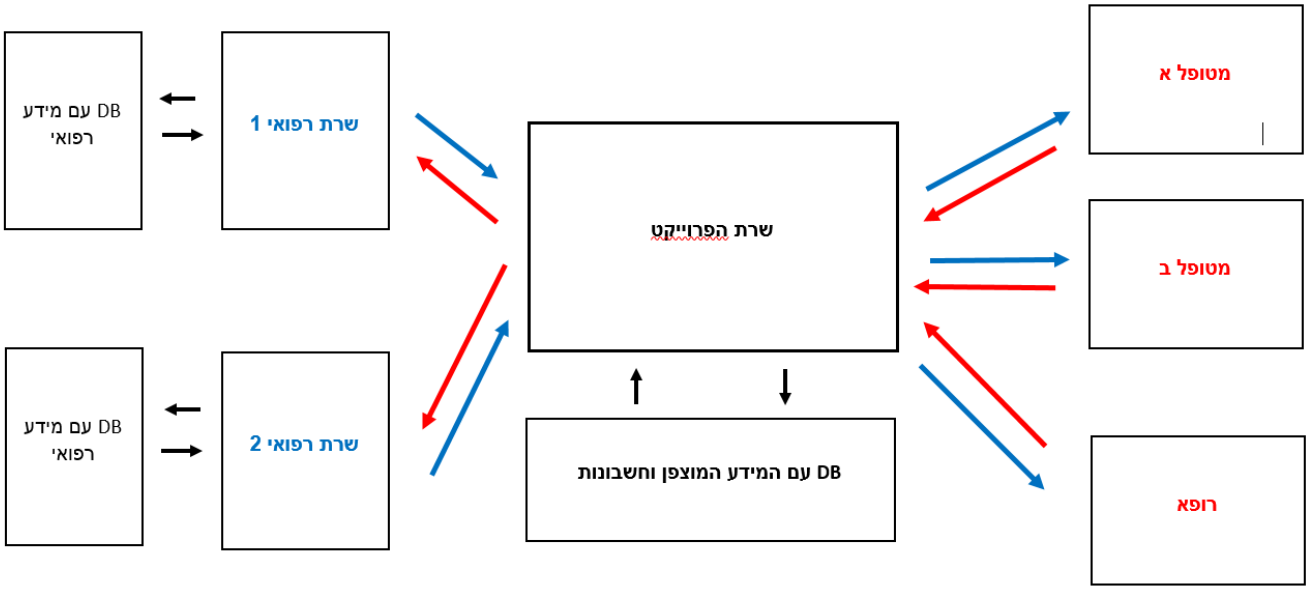
1. work out twofish encryption -> became AES (after we already bealt the TwoFish interactor)
2. create ECC encryption
3. send basic encrypted messages
4. create user databases and managers
5. medical institutes adding files
6. create login / signup sequence for patients
7. check if id is valid
8. create login / signup sequence for doctors
9. check registration of doctors
10. data requests from specific institutes
11. work on sending files
12. learn to send mails with python.
13. find best way to divide password
14. work out best way to send password parts
15. create forgot password function
16. create doctor requests for files
17. learn Flutter graphics and UI
18. create basic UI / GUI/ UX
19. reading / displaying files
20. tagging and organising files based on labels
21. create more advanced interface
22. Building company code
23. Dos attack defense
24. add function to delete users
25. update document
26. learn about HIPAA
27. implement and abide by HIPAA

אבטחה

כל התקשורת שלנו מתנהלת בסוקטים בעזרת פרוקוטול TCP- שהוא מנם אמין אבל לא מאובטח.
מתוקף המחסור בזמן הדבר היחיד שעובר בתקשורת כשהוא מוצפן לחלוטין זה הקבצים, והם גם שמורים מוצפנים בשרת. הם פתוחים רק בצד בית חולים ובצד לקוח והדרך היחידה לפתוח אותם היא בעזרת הסיסמא שמצפינה את המפתח. כל אחד יכול לשלוח בקשה לקובץ ולקבל אותם. אבל ההצפנה שבחרנו (AES) היא הצפנה טובה ואמינה. ולכן הקבצים שהם יקבלו יהיו חסרי תועלת.
גם העברת המפתחות ממטופל \ רופא לבית חולים מוצפנת לחלוטין בעזרת הצפנת ECC שהיא הצפנה מבוססת עקומים אליפטים והיא נחשבת הצפנה בטוחה מאוד.
לצורך השיחזור סיסמא נשלח קוד אימות לאימייל, כל שכל עוד אין לך גישה לאימייל של החשבון או לשרת שלנו, אין איך לשחזר את הסיסמא עם כוונות זדון.
הבהרה – כל המידע שהלקוח ממלא בהתחלה הוא לא מוצפן, והוא משמש בשרת לצורך יצירת החשבון, גם שמות הקבצים לא מוצפנים כדי שהשרת יוכל להראות למשתמש איזה קבצים שלו שמורים בשרת והמטופל יכול לקבל.
כמו כן, נבדקות מילות מפתח בכל מה מידע שאמור להישמר בשרת כדי לבדוק שזה לא קוד זדוני שישבש את הפעולות שלו.

שרת לקוח:

המטופל \ הרופא הם הלקוח של השרת, אבל כשהשרת מתקשר עם השרת בית חולים הוא עושה את זה במודל של שרת לקוח גם, כשהשרת שלנו מתפקד כלקוח.
השרת מאזין ומחכה לבקשות מהלקוח אבל הוא שומר אותו – כלומר הוא בקשר איתו גם בין בקשות, עד שהלקוח (המטופל \ רופא) מתנתק.
השרת בית חולים מאזין ומחכה לבקשות מהשרת אבל בין בקשה לבקשה הוא מנתק את התקשורת.
כלומר: לקוח - שרת



הצפנות:

ECC
(Elliptic Curve Cryptography=)

שיטת הצפנה אסימטרית (כלומר הצפנה שבו מפתח ההצפנה שונה ממפתח הפענוח. כלומר, כל משתמש מכין לעצמו זוג מפתחות :מפתח ציבורי שהוא מפתח הצפנה הנגיש לכל ומפתח פרטי מתאים, הנשמר בסוד ומשמש לפענוח. ההתאמה היא חד-חד-ערכית - לכל מפתח ציבורי קיים אך ורק מפתח פרטי יחיד המתאים לו, ולהפך. כדי להצפין מסר בשיטה זו על המצפין להשיג לידיו עותק אותנטי של המפתח הציבורי של המקבל, שבעזרתו הוא מצפין ושולח לו את המסר. רק המקבל מסוגל לשחזר את הטקסט המוצפן בעזרת המפתח הפרטי המתאים שברשותו. ביטחון שיטת המפתח הציבורי נשען על הקושי שבחישוב המפתח הפרטי מתוך המפתח הציבורי. מסיבה זו מכונה שיטה זו "א-סימטרית", בניגוד לשיטת הצפנה סימטרית ,שבה מפתח הפענוח זהה למפתח ההצפנה (או ניתן לחישוב בקלות מתוך מפתח ההצפנה).)

המונח הצפנת מפתח ציבורי מתייחס לשיטות קריפטוגרפיות שהדגש בהן הוא שימוש במפתח הצפנה לא סודי. למפתח זה יש שלל שימושים אבל אנחנו השתמשנו בו בשביל שיתוף מפתחות: בפרוטוקול שיתוף מפתח השולח והמקבל מחליפים ביניהם מידע כלשהו בערוץ פתוח הנגיש לכול ובתום חילופי המסרים המשתתפים חולקים ביניהם מידע סודי, כך שאיש לא יודע מהו, ממנו המשתתפים מכינים מפתח הצפנה , במקרה שלנו – מפתחות של בשביל הצפנת AES.

ECC עושה שימוש במבנה האלגברי-גאומטרי הנקרא עקום אליפטי מעל שדה סופי גדול, למימוש מערכת כגון פרוטוקול דיפי-הלמן או צופן אל-גמאל . הסבר יותר מפורט יעשה בעל פה – כי זה יותר מדי מסובך להסביר את זה בכתב.

AES

(Advanced Encryption Standard =)

קיצור ל"תקן הצפנה מתקדם" – צופן סימטרי (אלגוריתם הצפנה שבו משתמשים במפתח הצפנה יחיד הן להצפנה של הטקסט הקריא והן לפענוח של הטקסט המוצפן. הסיבה שהצופן נקרא סימטרי היא כי נדרש ידע שווה של חומר סודי (מפתח) משני הצדדים.)

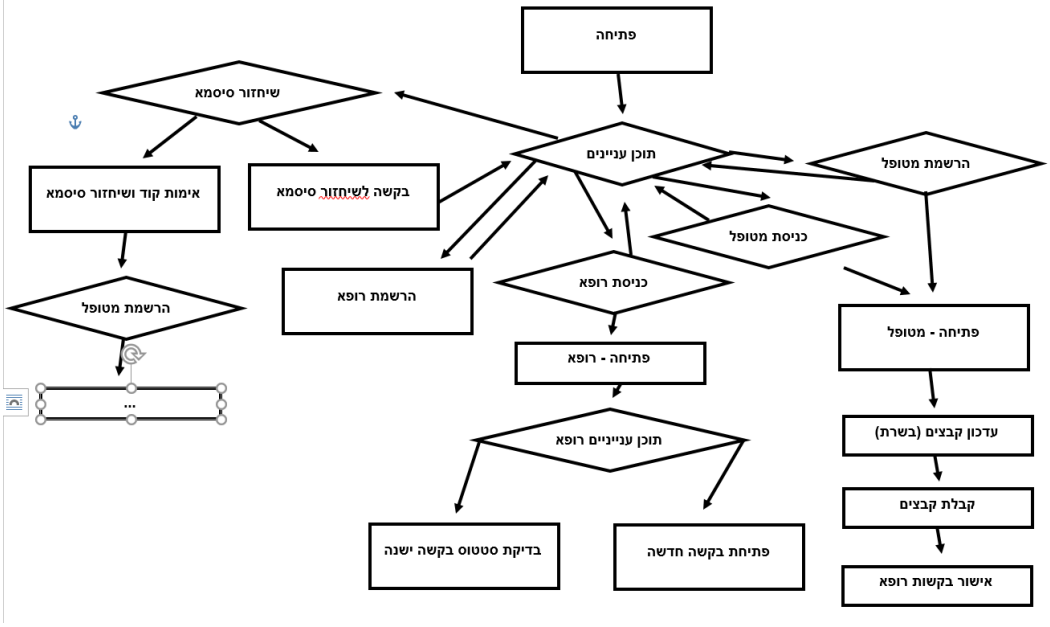
AES הוא צופן בלוקים, הוא לוקח את המידע שרוצים להצפין (שחייבת להיות בגודל קבוע – כלומר אם הגודל של המידע שרוצים להצפין לא מתחלק בגודל הבלוק, שבמקרה הזה הוא 16, צריך להפוך אותו לכזה (ע"י padding)) מחלק אותו לבלוקים בגודל מסויים. צופן בלוקים מקבל בלוק של המידע שרוצים להצפין ומפתח הצפנה סודי (שאנחנו מפיקים ע"י הצפנת ECC) ומפיק בלוק של טקסט-מוצפן. כאשר תוצאת הטרנספורמציה נקבעת על ידי מפתח ההצפנה . פענוח מתבצע באופן דומה, אלגוריתם הפענוח מקבל בלוק סיביות טקסט-מוצפן והמפתח שאיתו הוצפן ומחזיר את בלוק הסיביות המקורי. צופן בלוקים הוא מרכיב קריטי כמעט בכל מערכת הצפנה מודרנית.

עוד קצת עליו:

הצופן אומץ על ידי המכון הלאומי לתקנים וטכנולוגיה של ארצות הברית כתקן הצפנה רשמי שהתקבל בעולם כולו, להצפנת נתונים מאסיבית. השם המקורי של הצופן הוא ריינדל, והוא פותח על ידי הקריפטוגרפים הבלגיים יוהאן דאמן ווינסנט ריימן והוצע במהלך פרויקט בחירת התקן שאורגן על ידי NIST בשנת 2000. לאחר שזכה בתחרות אומץ על ידי ממשלת ארצות הברית באופן רשמי להצפנת נתונים מסווגים עבור. אלגוריתם AES נמצא בשימוש מעשי נרחב בכל העולם הן בתוכנה והן בחומרה וידוע כאלגוריתם בטוח.

זהו הצופן הסימטרי הפומבי הראשון שקיבל את אישור הסוכנות לביטחון לאומי האמריקאי כראוי להצפנת נתונים המוגדרים ברמת סיווג SECRET וכן TOP SECRET עבור ממשלת ארצות הברית, אם נעשה בו שימוש כחלק ממודול הצפנה מאושר. כלומר – הוא בטוח זה הצפנה טובה, ומי שיקבל את המידע ששמור לנו בשרת לא יוכל לעשות איתו כלום. אם זה מספיק טוב בשביל ה- NSA זה מספיק טוב בשבילי.

הוראות שימוש



בעיקרון זה לא מסך, זה פשוט הודעות ואפשרויות לקלט שמודפסות אחת אחרי השניה.

פתיחה - > הודעה על הצלחת חיבור לשרת (הודעת שלום)

תוכן עניינים - > יש 5 אפשרויות בחירה:

1. הרשמה של מטופל למערכת – אם מצליח מעביר לפתיחה של מטופל ואם לא חוזר לתוכן עניינים
2. הרשמה של רופא למערכת - גם אם מליח וגם אם נכשל מעביר לתוכן עיניינים, ההבדל הוא מה שקורה בשרת שבו נוסף למטופל את השדות של הרופא אם מצליח
3. כניסת מטופל (=התחברות של מטופל קיים) - אם מצליח, כלומר, יש מטופל כזה והסיסמא נכונה מעביר לפתיחה של מטופל ואם לא חוזר לתוכן עניינים
4. כניסת רופא - אם מצליח, כלומר, יש רופא כזה והסיסמא נכונה מעביר לפתיחה של רופא ואם לא חוזר לתוכן עניינים
5. שיחזור סיסמא: יש בחירה בין שתי אפשרויות :
1. בקשה חדשה – הודעה אם קוד אימות נשלחת לאימייל שמקושר למטופל. חוזרים אחריה לתוכן עניינים
2. אישור בקשה - המטופל מכניס את הקוד שקיבל באימייל – אם הכל נכון אז הוא מכניס סיסמא חדשה ומפתח מוצפן עי סיסמא חדשה נשלח לשרת. אם לא הוא חוזר לתוכן עניינים

פתיחה – מטופל - > הודעה על הצלחת התחברות של מטופל

עדכון קבצים -> נשלחת הודעה אוטומטית לשרת לעדכון קבצים שבבית חולים והם נשלחים מוצפנים לשרת.

קבלת קבצים - > הלקוח יכול להחליט איזה קבצים מהשרת הוא רוצה. וזה נשמר אצלו במחשב.

אישור בקשות רופא - > בודק אם יש בקשות של רופאים ואם כן שואל את המטופל על כל רופא האם לאשר את הבקשה שלו ואיזה קבצים לשלוח לו.

פתיחה – רופא - > הודעה על הצלחת התחברות של רופא

תוכן עניינים רופא -> נפתח תוכן עניינים שבו שתי אפשרויות:

1. שליחת בקשה חדשה למטופל בשביל קבצים שלו
2. בדיקת סטטוס של בקשה של מטופל – אם הוא אישר הרופא מקבל את הקבצים שהמטופל אישר וזה נשמר במחשב שלו.

אין כפתורים או תיבות טקסט – פשוט יש זרימה של קלט פלט.

עבור כל חלק שיש ללקוח, כלומר, המטופל או הרופא יש חלק תואם בשרת שנועד לספק את הבקשה. הלקוח שולח בקשה והשרת מטפל בה. אין פונקציות בשרת שפועלות שלא כתוצאה מבקשה כלשהיא של לקוח. חוץ מ- main שמתחילה את הריצה של השרת וגורמת לו לפעול.

חוץ מזה, יש גם את השרת בית חולים שיש לו 3 תפקידים: ליצור מפתח AES משותף עם הלקוח בעזרת ECC. במקרה של שיחזור סיסמא לשלוח ללקוח באופן מוצפן (גם ECC) את המפתח שלו בשביל שיצור סיסמא חדשה. להצפין את הקבצים ששייכים למטופל בעזרת המפתח AES המשותף ולשלוח אותם לשרת.

אם כל חלק של בקשה הולך תקין מודפסת הודעה. על כל אפשרות של לקוח הוא מקבל הודעה אם הפעולה הצליחה או לא. במקרה של בקשות קבצים, נוצרת תיקיה בשם של הת.ז. של הלקוח בתיקיה client_files של הפרוייקט.

הפרוייקט הוא מאוד אינטואיטיבי, בכל חלק של הפרוייקט מודפסות הודעות שמסבירות בדיוק מה קורה ומה צריך לעשות.

כל האלמנטים של המטופל נבדקים בהרשמה שלו: ישנה בדיקה של ספרת הביקורת בת.ז. הפלאפון חייב להיות מורכב ממספרים מלבד באורך ספציפי, הסיסמא חייבת להיות באורך של לפחות שש תווים ועד ארבעים ולהכיל אותיות ומספרים.

בסיס נתונים – טבלאות

מסד נתונים בשרת

USERS

הבהרה: אם השיוך הוא של רופא, זה אמנם לא חובה. אבל רופא שנרשם חייב שיהיה לו אותם

שם העמודה	סוג	חובה	שיוך
ת.ז. – מפתח ראשי	TEXT	כן	מטופל
שם	TEXT	כן	מטופל
אימייל	TEXT	כן	מטופל
ת.ז. של איש קשר למקרה חירום	TEXT	כן	מטופל
מפתח מוצפן ע"י סיסמא	BLOB	כן	מטופל
טלפון	TEXT	כן	מטופל
בדיקת התחברות	BLOB	כן	מטופל
רופאים שביקשו מידע	TEXT	לא	מטופל
בדיקת שיחזור סיסמא	BLOB	לא	מטופל
עדכון מידע אחרון	TEXT	לא	מטופל
מפתחות של לקוח	TEXT	לא	מטופל
מקום עבודה	TEXT	לא	רופא
התמחות	TEXT	לא	רופא
מספר רשיון	TEXT	לא	רופא

FILES

שם העמודה	סוג	חובה
ת.ז.	TEXT	כן
שם קובץ	TEXT	כן
קובץ (מוצפן)	BOLB	כן

מסד נתונים בשרת בית חולים

USERS

שם העמודה	סוג	חובה
ת.ז. – מפתח ראשי	TEXT	כן
מפתח הצפנה	TEXT	כן

מדריך התקנה

בעיקרון פשוט צריך את התיקיה client_files וסביבה שמריצה קוד פייתון, מריצים את הקוד basic_client וזה יתחיל להעלות את הזרימה של הקלט פלט.

התאמת סביבת עבודה להרצה

בשביל שהפרויקט ירוץ באמת צריך שלפרויקט שלהו יהיה IP, כרגע הכי קרוב לזה, זה מחשבים באותה רשת Wi-Fi . השרת רץ על פורט 7239 ומשתמש בפרוטוקול TCP בתקשורת מבוססת סוקטים.

כמו כן, צריך שרת של בית חולים שהאפליקציות שכתבנו ישולבו ויותאמו אליו. מה שלא סביר שיקרה אי-פעם.

נתונים של משתמש לדוגמא כפי שהם שמורים בשרת: (מה שצמוד זה שדות ריקים או מלאים מידע בינארי:

325002426	zlata brun	zlatabrun@gmail.com	310031620		0585999244	qwe.qwe	meuhedet	child				1587449480.966588

שם משתמש: 325002426

סיסמא: qwe234

אין באמת מנהל: זה רץ לבד. יש רק משתמשים רגילים. אבל בשביל לשנות משהו בשרת צריך לגשת לשרת ולשנות ידנית בקוד או לגשת ידנית לבסיס נתונים.

מדריך למפתח

קובץ: client_files/__init__

הקובץ הזה מאפשר לגשת גם לקוד שכתוב קבצים אחרים וכך לפצל את קוד הפרוייקט בין כל מיני תקיות. פשוט כותבים:

From <file in this directory> import <* \ name of func>

אם כותבים * כל הקוד שבקובץ ייבוא, אם לא רק הפונקציה \ מחלקה תיובא.

אח"כ כשרוצים להשתמש בפונקציה מקובץ אחר פשוט רושמים:

<directory_name>.<the_the_func_you_want_to_use>(<func_parameters>)

במקרה הזה:

client_files.<the_the_func_you_want_to_use>(<func_parameters>)

קובץ: client_files/basic_client

הקובץ הזה הוא חלק מהקבצים שנמצאים בצד לקוח של הפרוייקט והם מה שבעצם מתקינים כדי לעבוד עם הפרוייקט.

הקובץ הזה מכיל את ה- main של הלקוח ואותו מריצים כשרוצים להתחיל.

לפני ה- main מופיעים שלושה משתנים: ה-IP , שמקרה ותהיה לנו כתובת IP נורמלית לשרת זה ישונה לזה, כרגע זה "127.0.0.1" כלומר גם השרת וגם הלקוח רצים על אותו מחשב. ה-PORT שהשרת מאזין בו ומשתנה בשם MSG_SERVER שהוא פשוט קידומת להודעות של השרת, כשמוסיפים לו את האופציה שהלקוח בוחר בתוכן עיניינים זה אומר לשרת איזה אופציה המשתמש בחר ואיזה אופציה להריץ בתגובה ואיך לענות על הבקשה.

לפרוט על סוגי ההודעות ראו מסמך RequestCode.

כשמריצים את הקוד, במקרה של הצלחה של חיבור אם השרת, נפתחות אופציות של הרשמה והתחברות, המשתמש מכניס קלט של בחירת אופציה, שנשלחת לפונקציה

Client_files.option_manager_client()

שלא נמצאת באותו קובץ, אם מוחזר מהפונקציה 512 למשתנה msg_to_server אז זה אומר שהמשתנים (key – המפתח הצפנה, c_id – הת.ז. של הלקוח ו- is_dr – האם ההתחברות היא של רופא או לקוח) מכילים מידע ולא "0" ושאפשר לעבור לשלב הבא. ואפשר לעבור לשלב הבא, אם לא התוכן עניינים פשוט יחזור על עצמו עד שהתשובה ב- msg_to_server.

server_msg הוא משתנה שמכיל את התשובות שהתקבלו בסוקט מהשרת.

אם האופציה שנבחרה (m) לא שווה לשמונה- כלומר הלקוח רוצה להתנתק אז זה בודק האם ההתחברות היא של לקוח או רופא ולפי זה מפנה אותם לאפשרויות או של לקוח מחובר או רופא מחובר.

קובץ: client_files/client_register_options

הקובץ הזה הוא חלק מהקבצים שנמצאים בצד לקוח של הפרוייקט והם מה שבעצם מתקינים כדי לעבוד עם הפרוייקט.

הקובץ הזה מכיל את כל הפונקציות שאחראיות להתחברות \ הרשמה ולקבלת מפתח הצפנה.

1. Recreate_password()
הפונקציה הזו מקבלת את הסוקט שאיתו מתקשרים עם השרת ופותחת תוכן עיניינים:
האופציה הראשונה זה לשלוח בקשה לשרת לשלוח אימייל שמכיל קוד לאימות סיסמא, מכניסים id וזה נשלח לשרת שם הוא שולח אימייל לכתובת שמשוייכת לת.ז. הזה ב-DB.
האופציה השנייה היא אחרי שקיבלת קוד באימייל, להכניס אותו ותז. לשרת, אם הקוד נכון אז אתה צריך להכניס סיסמא חדשה. ואז אתה חוזר שוב לתוכן עיניינים ויכול להתחבר עם הסיסמא כרופא או כלקוח.
2. Patient_signup()
הרשמה כלקוח, אתה ממלא את הנתונים שהשרת צריך להרשמה (תז. , סיסמא, אימייל , ת.ז. איש קשר למקרה חירום, פלאפון) אם הבקשה שנשלחה לשרת תקינה מתחילה העברת מפתחות ECC בין השרת בית חולים (דרך השרת הפרוייקט) ונוצר מפתח, מצפינים אותו ע"י הסיסמא ומצפינים בו מסר ושתיהם נשלחים לשרת. אם הכל עובר תקין ה id וה- key נשלחים יחד עם הקוד "512" והלקוח יעבור למסך הבא
3. Doctor_signup()
הרשמה כרופא, אתה ממלא את הנתונים שהשרת צריך להרשמה (מספר רשיון רופא, מקום עבודה, התמחות) מודפסות הודעות במקרים של תקלה או הצלחה אבל התשובה היא אותו דבר וכדי להיכנס לחשבון הרופא יצטרך לבחור doctor_logint()
4. Patient_login()
המשתמש ממלא ת.ז. וסיסמא, השרת בודק שיש משתמש כזה, השרת שולח את המפתח המוצפן ואת המסר הוצפן, פותחים בעזרת הסיסמא את המפתח, אם הוא מצליח לפתוח את המסר ("HELLO") אז סימן שהסיסמא נכונה והמפתח תקין ו ה id וה- key נשלחים יחד עם הקוד "512" והלקוח יעבור למסך הבא
5. Doctor_login()
המשתמש ממלא ת.ז. וסיסמא, מתבצעת בדיקה האם יש רופא כזה, השרת שולח את המפתח המוצפן ואת המסר הוצפן, פותחים בעזרת הסיסמא את המפתח, אם הוא מצליח לפתוח את המסר ("HELLO") אז סימן שהסיסמא נכונה והמפתח תקין ו ה id וה- key נשלחים יחד עם הקוד "512" והרופא יעבור לשלב הבא, אבל בשינוי – is_dr יהיה שווה ל- TRUE והרופא יעבור למסך התחברות של הרופאים.

קובץ: client_files/decrypter
קובץ: hospital_files/ decrypter
הקובץ זהה בשתיהם – אחראי על פענוח של הצפנת AES והחלפת מפתחות ECC

יש את הפונקציה pad שאם הגודל של מה שרוצים להצפין לא מתחלק ב16 מוסיפה "\0" עד שזה בגודל המתאים, כי אחרת ההצפנה לא תעבוד.

חוץ מזה יש את המחלקה decrypter שיש בה את כל הפונקציות שצריך בשביל הצפנה ופענוח של הצפנת AES והחלפת מפתחות ECC + משתנה של המפתח

המחלקה ישכולה לשמש גם כ Encrypter. יש לה יכולת גם להצפין למרות שמה. ההבדל היחיד במחלוקת הוא בתפקידם בהחלפת מפתחות ה- ECC.

בתוכה יש את הפונקציות:

1. __init__ בשביל ליצור את המחלקה
2. Set_public_data_from_encrypter_and_get_own_data()
3. הפונקציה מקבלת מידע מה – Encrypter ומשתמשת בו כדי למצוא את הנקודה R והמפתח של עצמה. לוקחת את ה- X של הנקודה, עודה לזה hash ויוצרת מפתח AES. לפי פרוטוקול ההחלפת מפתחות של ECC.
4. Encrypte()
הפונקציה משקבלת מסר ובעזרת המפתח ששמור במחלקה מצפינה אותו ומחזירה אותו מוצפן.
5. Encrypte_file()
הפונקציה קוראת מידע מקובץ ושולחת את המידע לפונקציה Encrypte()
6. Decrypte()
הפונקציה מקבלת מידע מוצפן ובעזרת המפתח ששמור במחלקה מפענחת ומחזירה אותו.
7. Decrypte_file()
הפונקציה מקבלת מידע מוצפן, השם של הקובץ, ו- path לשמור אותו, שולחת את המידע לdecrypte מקבלת אותו מפוענח, ושומרת בpath את הקובץ ע"י כתיבה של המידע המפוענח לתקו ושינוי השם שלו.

קובץ: client_files/encrypter
קובץ: hospital_files/ encrypter
הקובץ זהה בשתיהם – אחראי על פענוח של הצפנת AES והחלפת מפתחות ECC

יש את הפונקציה pad שאם הגודל של מה שרוצים להצפין לא מתחלק ב16 מוסיפה "\0" עד שזה בגודל המתאים, כי אחרת ההצפנה לא תעבוד.

חוץ מזה יש את המחלקה encrypter שיש בה את כל הפונקציות שצריך בשביל הצפנה ופענוח של הצפנת AES והחלפת מפתחות ECC + משתנה של המפתח

המחלקה יכולה לשמש גם כ Decrypter. יש לה יכולת גם לפענח למרות שמה. ההבדל היחיד במחלוקת הוא בתפקידם בהחלפת מפתחות ה- ECC.

בתוכה יש את הפונקציות:

1. __init__ בשביל ליצור את המחלקה
2. Get_data_to_send_publicly()
הפונקציה מקבלת את המפתח הציבורי ששמור ב- Encrypter ("we will rock you" – לא חשוב)
3. Set_r()
הפונקציה מקבלת מידע מה – decrypter ומשתמשת בו כדי למצוא את הנקודה R והמפתח של עצמה. לוקחת את ה- X של הנקודה, עודה לזה hash ויוצרת מפתח AES. לפי פרוטוקול ההחלפת מפתחות של ECC.
4. Encrypte()
הפונקציה משקבלת מסר ובעזרת המפתח ששמור במחלקה מצפינה אותו ומחזירה אותו מוצפן.
5. Encrypte_file()
הפונקציה קוראת מידע מקובץ ושולחת את המידע לפונקציה Encrypte()
6. Decrypte()
הפונקציה מקבלת מידע מוצפן ובעזרת המפתח ששמור במחלקה מפענחת ומחזירה אותו.
7. Decrypte_file()
הפונקציה מקבלת מידע מוצפן, השם של הקובץ, ו- path לשמור אותו, שולחת את המידע לdecrypte מקבלת אותו מפוענח, ושומרת בpath את הקובץ ע"י כתיבה של המידע המפוענח לתוכו ושינוי השם שלו.

קובץ: client_files/doctor_options

הקובץ הזה הוא חלק מהקבצים שנמצאים בצד לקוח של הפרוייקט והם מה שבעצם מתקינים כדי לעבוד עם הפרוייקט.

ישנם 3 פונקציות שהן מה שהרופא יכול לעשות אחרי ההתחברות שלו:

1. New_date_request()
הרופא מתבקש להכניס תז. של מטופל, אם היא תקינה הוא יוצר מפתח עם הבית חולים תחת הת.ז. שלו + ת.ז. מטופל כדי שהמטופל יוכל להשתמש בו כדי להצפין את הקבצים שהוא מאשר. (הבקשה נשמרת אצל השרת)
2. Request_status()
הופונקציה מבקשת להכניס ת.ז. של המטופל שרוצים לבדוק , אם המטופל אישר את בקשת הרופא, אז הרופא יקרא יקבל את הקבצים ששמורים תחת הת.ז. שלו ושל המטופל ביחד ויקבל את המפחת שהוא יצר ב- New_data_request
3. Get_and_save_file_doctor()
פונקציה שמקבלת מהשרת את הקובץ שמבוקש – במקרה זה קובץ של מטופל שאישר את בקשת הרופא.

קובץ: client_files/patient_options

הקובץ הזה הוא חלק מהקבצים שנמצאים בצד לקוח של הפרוייקט והם מה שבעצם מתקינים כדי לעבוד עם הפרוייקט.

ישנם 5 פונקציות שהן מה שהמטופל יכול לעשות אחרי ההתחברות שלו:

1. Start_data_update()
הפונקציה שולחת בקשה לשרת לבדוק אם יש קבצים בשרת בית חולים שאין אצלו ואם לא אז לקבל אותם
2. Get_and_save_file ()
פונקציה שמקבלת מהשרת את הקובץ שמבוקש
3. Send_key_to_new_hospital()
במקרה שיש בית חולים חדש שיש קבצים של הלקוח שהוא רוצה, המשתמש שולח את המפתח הקיים שלו כדי שהוא ישמור אותו ויצפין בהם את הקבצים שלו.
4. Check_doctor_request()
5. הפונקציה בודקת אם בשרת שמורים בקשות מרופאים, אם כן היא עוברת אחד אחד ושואלת את הלקוח איזה רופאים לאשר. את אלה שהוא אישר היא שולחת לפונקציה Approve_doctor_request()
6. Approve_doctor_request()
הפונקציה מקבלת id של רופא שהמטופל אישר אותו ושואלת איזה קבצים לאשר לו לקבל. היא מצפינה אותם תחת ה-id של הרופא והמטופל מחוברים ושולחת אותם לשרת.

קובץ: client_files/keys_handler

הקובץ הזה הוא חלק מהקבצים שנמצאים בצד לקוח של הפרוייקט והם מה שבעצם מתקינים כדי לעבוד עם הפרוייקט.

ישנם 3 פונקציות שמטפלות בתקשורת שקשורה למפתחות ECC:

1. Get_new_key()
הפונקציה משמשת כדי ליצור מפתח חדש עם שרת בית החולים דרך שרת הפרוייקט
2. Get_old_key()
הפונקציה משמשת כדי לקבל את המפתח הישן של הלקוח משרת בית החולים דרך שרת הפרוייקט.
3. Send_old_key()
הפונקציה משמשת כדי לשלוח לשרת בית חולים חדש את המפתח הקיים של הלקוח.

קובץ: client_files/option_manager_client_side

הקובץ הזה הוא חלק מהקבצים שנמצאים בצד לקוח של הפרוייקט והם מה שבעצם מתקינים כדי לעבוד עם הפרוייקט.

יש בו רק פונקציה אחת ()option_manager_client שתפקידה לקבל את הבקשה שהלקוח בחר בתוכן עיניינים הראשי ולבחור לפי זה את הפונקציה המתאימה לבקשה שלו.

קובץ: server_files/__init__

הקובץ הזה מאפשר לגשת גם לקוד שכתוב קבצים אחרים וכך לפצל את קוד הפרוייקט בין כל מיני תקיות. פשוט כותבים:

From <file in this directory> import <* \ name of func>

אם כותבים * כל הקוד שבקובץ ייבוא, אם לא רק הפונקציה \ מחלקה תיובא.

אח"כ כשרוצים להשתמש בפונקציה מקובץ אחר פשוט רושמים:

<directory_name>.<the_the_func_you_want_to_use>(<func_parameters>)

במקרה הזה:

server_files.<the_the_func_you_want_to_use>(<func_parameters>)

קובץ: server_files/basic_server

התיקיה הזו ממכילה את הקבצים שקשורים לריצה של השרת של הפרוייקט.

הקובץ הזה בעצם מכיל את ה- main של השרת ואחראי להאזין למשתמשים חדשים.

Num הוא מספר הבקשות שהשרת יקבל לפני לפני שיסגר.

על כל בקשה שמגיע לשרת הוא יוצר thread חדש של הפונקציה server_files.client_handler() שמטפלת בלקוח, כלומר הוא יכול לטפל בבקשות קיימות ועדיין להמשיך להאזין, לחכות ללקוח חדש ולטפל בו.

קובץ: server_files/new_comminication

התיקיה הזו ממכילה את הקבצים שקשורים לריצה של השרת של הפרוייקט.

הקובץ מכיל את הפונקציה client_handler() שמקבלת socket של לקוח חדש, ופשוט כל בקשה חדשה שצצה ללקוח היא שולחת לפונקציה server_files.option_manager() שמנתבת אותם, וכל עוד הלקוח לא מכבה את הצד שלו, או שולח "508" כלומר הוא רוצה להתנתק היא ממשיכה לרוץ.

קובץ: server_files/option_manager

התיקיה הזו ממכילה את הקבצים שקשורים לריצה של השרת של הפרוייקט.

יש בו רק פונקציה אחת option_manager() שתפקידה לקבל את הבקשה שהלקוח שלח ולבחור לפי זה את הפונקציה המתאימה לבקשה שלו. (כל בקשה מתחילה בספרות 5XX)

קובץ: server_files/database_interactor

התיקיה הזו ממכילה את הקבצים שקשורים לריצה של השרת של הפרוייקט.

הקובץ הזה אחראי לתקשורת של השרת עם ה- SQLite מסד נותנים.

יש בו בהתחלה מחלקות שבעיקר אחראיות לחבר את סוג הבקשה לשם המדוייק שלה בשרת כדי שזה יעבוד באופן יותר אינטואיטיבי וע"י יצירת מחלקה כזו תוכל למצוא יותר בקלות את שמות העמודות בטבלאות. יש גם פונקציה בשם is_safe_save_for_query() שבודקת אם המידע שהמשתמש מכניס לשרת לא מכיל קובץ זדוני שישבש את הפילות שלו ע"י סריקה של מספר מילות מפתח.

הערה: בכל מקום שרשום שהיא שומרת אותם במסד נתונים זה תמיד אחרי שימוש בפונקציה שבודקת את הבטיחות של הקלט.

כמו כן יש את המחלקה שאחראית לתקשר עם ה- DB, היא מכילה את הפונקציות הבאות:

1. __init__ בשביל ליצור את המחלקה, במקרה ואין DB היא יוצרת אותו ואם כן היא פשוט יוצרת קישור אליו
2. Add_user()
אם אין כבר בטבלה משתנה עם הת.ז. שבמידע שהיא מקבלת וכל המידע תקין היא מוסיפה אותו בעזרת המילון שהיא מקבלת שמכיל את המידע בשביל יצירת המשתנה.
3. User_exist()
הפונקציה בודקת האם המשתמש אם התז. שהיא מקבלת קיים בשרת
4. Get_user_data()
הפונקציה מקבלת את סוג המידע המבוקש ותז. שולחת את המידע שנמצא בעמודה המתאימה בשורה שמתאימה לתז. במסד נתונים
5. Add_file()
הפונקציה מקבלת קובץ (מוצפן), ת.ז. ושם ושומרת אותם במסד נתונים.
6. Get_file()
הפונקציה מקבלת תז. ושם קובץ ומחזירה את הקובץ שהוא הצלבה של שניהם, כי אין מפתח ראשי בטבלת מסמכים, ככה שגם התז. וגם שם הקובץ יכולים להופיע מבפר פעמים.
7. Add_doctor()
הפונקציה מקבלת 3 פרמטרים שיכולים להוסיף לעמודות של משתמש רגיל שהוא מטופל: מספר רשיון רפואי, מקום עבודה והתמחות. ואם הוא לא רופא והכל תקין היא שומרת אותם בשורה שמתאימה לתז. שקיבלה.
8. Is_a_doctor()
הפונקציה בודקת האם למשתמש שמקושר לת.ז. שהיא מקבלת בטבלה יש רשיון רפואי – אם כן זה אומר שהוא כבר נרשם והוא רופא.
9. Update_user()
הפונקציה מקבלת תז. של משתמש , מידע, וסוגו ומעדכנת אותו במסד נתונים לאחר שורת בדיקות תקינות.

קובץ: server_files/patient_options

התיקיה הזו ממכילה את הקבצים שקשורים לריצה של השרת של הפרוייקט.

הקובץ הזה מכיל את הפונקציות שמטפלות בלקוח מסוג מטופל בשרת

הערה: בכל פונקציה השרת יוצר מחלקה של אינטרקטור שמתקשרת עם ה- DB.

1. Recreate_password_a()
השרת מקבל ת.ז. ומוציא מה - DB את האימייל שקשור לת.ז. הוא יוצר מספר רנדומלי בטווח של 10000-99999 שול אותו לאימייל שהוציא ושומר את הקוד ב-DB בעמודה שקשורה לת.ז.
2. Recreate_password_b()
השרת מקבל ת.ז. וקוד , הוא בודק אם הקוד שמשוייך לת.ז. הזה ב-DB זהה לו. אם כן הלקוח והבית חולים יוצרים דרכו את האפשרות לקבל מפתח ישן, הלקוח שולח לו את המפתח מוצפן ע"י הסיסמא והשרת מעדכן את המידע במסד נתונים.
3. Patient_signup()
השרת מקבל נתונים של משתמש (תז. , סיסמא, אימייל , ת.ז. איש קשר למקרה חירום, פלאפון) אם הבקשה שנשלחה לשרת תקינה מתחילה העברת מפתחות ECC בין הלקוח לבית חולים דרך השרת הפרוייקט. השרת מקבל מפתח מוצפן ע"י סיסמא ומסר בדיקה מוצפן ומנסה לשמור את המשתמש ב-DB. ושולח את התוצואה (הצליח\לא הצליח) למשתמש.
4. Patient_login()
השרת בודק שיש מטופל שמקושר לת.ז. שהוא מקבל, ושולח את המפתח המוצפן ואת המסר הוצפן למטופל.
5. update_files()
הפונקציה מקבלת מהלקוח תז. יוצרת סוקט עם הבית חולים ומבקשת מהבית חולים לבדוק אם מאז העדכון האחרון (מידע שנמצא בdb) נוספו קבצים חדשים למטופל הזה. אם כן, שרת הבית חולים שולח לשרת אותם והם נשמרים ב- DB.
6. Get_patient_file()
הפונקציה מקבלת שם של קובץ ות.ז. ואם קיים קובץ כזה ב-DB שולחת אותו למשתמש.
7. Check_doctor_request()
השרת בודק האם השדה של בקשות מהרופאים בשורה של המטופל במסד נתונים לא ריק, אם הוא לא ריק היא שולחת אותו למטופל.
8. Save_file_for_doctor()
הפונקציה מקבלת מהלקוח קבצים אותם היא שומרת ב-DB תחת הת.ז. שהיא מקבלת (שהוא תז. של הרופא והמטופל מחוברים).

קובץ: server_files/doctor_options

התיקיה הזו ממכילה את הקבצים שקשורים לריצה של השרת של הפרוייקט.

הקובץ הזה מכיל את הפונקציות שמטפלות בלקוח מסוג רופא בשרת

הערה: בכל פונקציה השרת יוצר מחלקה של אינטרקטור שמתקשרת עם ה- DB.

1. Doctor_signup()
השרת מקבל נתונים של משתמש השרת מקבל נתונים של משתמש (מספר רשיון רופא, מקום עבודה, התמחות) אם הבקשה שנשלחה לשרת תקינה הוא מנסה לשמור את הנתונים כתוספת לנתונים של המטופל של המשתמש ב-DB. ושולח את התוצאה (הצליח\לא הצליח) למשתמש.
2. Doctor_login()
המשתמש ממלא ת.ז. וסיסמא, מתבצעת בדיקה האם יש רופא כזה, השרת שולח את המפתח המוצפן ואת המסר הוצפן, פותחים בעזרת הסיסמא את המפתח.
3. New_data_request()
השרת מקבל ת.ז. של רופא ושל המטופל שהוא רוצה לבקש ממנו מידע, הוא יוצר החלפת מפתחות בין הרופא לבית חולים ומוסיף את הבקשה של הרופא לרשימה של בקשות שקשורות למטופל ששמורות ב- DB.
4. Check_status()
השרת מקבל ת.ז. של רופא ותז. של המטופל שהוא רוצה לבדוק, אם במידע של המטופל במסד נתונים , הרופא נמצא עדיין ברשימה ויש לו שם "1" במקום 0, השרת יוצר לרופא דרך לקבל את המפתח ששמור בבית חולים תחת הת.ז. שלו ושל המטופל ביחד.

קובץ: server_files/encryption_func

התיקיה הזו ממכילה את הקבצים שקשורים לריצה של השרת של הפרוייקט.

הקובץ הזה מכיל את הפונקציות שקשורות בעיקר להחלפת מפתחות בין הלקוח לשרת בית חולים דרך השרת הזה:

1. exchange_keys_server()
השרת יוצר תקשורת עם הבית חולים ופשוט משמש צינור בין השרת בית חולים ללקוח, כל מה שהוא מקבל מהלקוח שולח לבית חולים ולהיפך. זה בשביל ליצור בינהם מפתח ECC.
2. Get_user_key()
כמו הקודמת רק אם תקשורת טיפה שונה, פה המטרה זה שהלקוח יקבל מפתח שכבר היה לו מהבית חולים בצורה מוצפנת ולא ליצור חדש.
3. Send_key_to_hospital()
כמו הקודמת, רק שפה המטרה היא להעביר לבית חולים חדש מפתח ישן ולא מהבית חולים ללקוח.
4. Text_to_binary()
לוקחת מחרוזת של מבנה בינארי ("b"") והופכת אותו לבינארי באמת
כלומר, הפונקציה עושה את ההיפך מ- str(<byte_type>)

קובץ: hospital_files/__init__

הקובץ הזה מאפשר לגשת גם לקוד שכתוב קבצים אחרים וכך לפצל את קוד הפרוייקט בין כל מיני תקיות. פשוט כותבים:

From <file in this directory> import <* \ name of func>

אם כותבים * כל הקוד שבקובץ ייבוא, אם לא רק הפונקציה \ מחלקה תיובא.

אח"כ כשרוצים להשתמש בפונקציה מקובץ אחר פשוט רושמים:

<directory_name>.<the_the_func_you_want_to_use>(<func_parameters>)

במקרה הזה:

hospital_files.<the_the_func_you_want_to_use>(<func_parameters>)

קובץ: hospital_files/hospital_side_code

הקבצים בתיקיה הזו אמורים לדמות שרת בית חולים שמכיל מידע על משתמש. בעולם מושלם שבו הפרוייקט שלנו באמת היה מופעל, הפונקציות שכתבנו היו מוטמעות בקוד של השרתים שלהם ולא היה צורך בבנייה של שרת.

הקובץ הזה בעצם מכיל את ה- main של השרת ואחראי להאזין למשתמשים חדשים.

Num הוא מספר הבקשות שהשרת יקבל לפני לפני שיסגר.

על כל בקשה שמגיע לשרת הוא יוצר thread חדש של הפונקציה hospital_files.request_handler() שמטפלת בלקוח, כלומר הוא יכול לטפל בבקשות קיימות ועדיין להמשיך להאזין, לחכות ללקוח חדש ולטפל בו.

קובץ: hospital_files/new_communication

הקבצים בתיקיה הזו אמורים לדמות שרת בית חולים שמכיל מידע על משתמש. בעולם מושלם שבו הפרוייקט שלנו באמת היה מופעל, הפונקציות שכתבנו היו מוטמעות בקוד של השרתים שלהם ולא היה צורך בבנייה של שרת.

הקובץ מכיל את הפונקציה request_handler() שמקבלת socket של השרת של הפרוייקט, ואת הבקשה שהיא מקבלת מהשרת היא שולחת לפונקציה hospital_files.option_manager() שמנתבת אותם, אחרי שהפונקציה מסיימת לרוץ היא מנתקת את הקשר עם השרת, כלומר זו פונקציה של בקשות חד פעמיות, אין תקשורת מתמשכת והמתנה לבקשות חדשות. בשביל זה השרת פרוייקט יצטרך ליצור תקשורת חדשה עם השרת בית חולים.

קובץ: hospital_files/hospital_option_manager

הקבצים בתיקיה הזו אמורים לדמות שרת בית חולים שמכיל מידע על משתמש. בעולם מושלם שבו הפרוייקט שלנו באמת היה מופעל, הפונקציות שכתבנו היו מוטמעות בקוד של השרתים שלהם ולא היה צורך בבנייה של שרת.

ש בו רק פונקציה אחת ()option_manager שתפקידה לקבל את הבקשה ששרת הפרוייקט שלח ולבחור לפי זה את הפונקציה המתאימה לבקשה שלו. (כל בקשה מתחילה בספרות 65X)

קובץ: hospital_files/hospital_database_interactor

הקבצים בתיקיה הזו אמורים לדמות שרת בית חולים שמכיל מידע על משתמש. בעולם מושלם שבו הפרוייקט שלנו באמת היה מופעל, הפונקציות שכתבנו היו מוטמעות בקוד של השרתים שלהם ולא היה צורך בבנייה של שרת.

הקובץ הזה אחראי לתקשורת של השרת עם ה- SQLite מסד נותנים.

יש בו בהתחלה מחלקות שבעיקר אחראיות לחבר את סוג הבקשה לשם המדוייק שלה בשרת כדי שזה יעבוד באופן יותר אינטואיטיבי וע"י יצירת מחלקה כזו תוכל למצוא יותר בקלות את שמות העמודות בטבלאות. יש גם פונקציה בשם ()is_safe_save_for_query שבודקת אם המידע שהמשתמש מכניס לשרת לא מכיל קובץ זדוני שישבש את הפילות שלו ע"י סריקה של מספר מילות מפתח.

הערה: בכל מקום שרשום שהיא שומרת אותם במסד נתונים זה תמיד אחרי שימוש בפונקציה שבודקת את הבטיחות של הקלט.

כמו כן יש את המחלקה שאחראית לתקשר עם ה- DB, היא מכילה את הפונקציות הבאות:

1. __init__ בשביל ליצור את המחלקה, במקרה ואין DB היא יוצרת אותו ואם כן היא פשוט יוצרת קישור אליו
2. ()Add_user
היא מקבלת ת.ז. ומפתח ואם אין כבר בטבלה משתנה עם הת.ז. שבמידע שהיא מקבלת וכל המידע תקין היא מוסיפה אותו ואת המפתח ל-DB.
3. ()Get_user_key
4. אם יש משתמש עם הת.ז. הזה היא שולפת את המפתח מהמסד נתונים ומחזירה אותו.

קובץ: hospital_files/server_options

הקבצים בתיקיה הזו אמורים לדמות שרת בית חולים שמכיל מידע על משתמש. בעולם מושלם שבו הפרוייקט שלנו באמת היה מופעל, הפונקציות שכתבנו היו מוטמעות בקוד של השרתים שלהם ולא היה צורך בבנייה של שרת.

הקובץ הזה מכיל את מה שמטפל בבקשה של השרת לעדכן קבצים. יש בו 2 פונקציות:

1. Get_files()
אם מאז הזמן האחרון שהפונקציה הזו רצה בשרת נוספו קבצים חדשים לתיקיה שהשם של הוא הת.ז. של משתמש הוא קורא ל- encrypte_files() מצפין אותם ושולח אותם לשרת.
2. Encrypte_files()
מקבלת רשימה של שמות קבצים ו-תז. , היא לוקחת הרשימה של הקבצים, נכנסת לpath של המיקום של התיקיה עם השם של התז. ומצפינה אותם, מחזירה אותה רשימה רק שהקבצים מוצפנים הפעם.

קובץ: hospital_files/ckient_through_server_options

הקבצים בתיקיה הזו אמורים לדמות שרת בית חולים שמכיל מידע על משתמש. בעולם מושלם שבו הפרוייקט שלנו באמת היה מופעל, הפונקציות שכתבנו היו מוטמעות בקוד של השרתים שלהם ולא היה צורך בבנייה של שרת.

הקובץ הזה מכיל את הפונקציות שמטפלות בבקשות של לקוח שעוברות דרך השרת פרוייקט. יש בו 2 פונקציות:

1. Exchanging_keys()
2. הפונקציה משמשת כדי ליצור מפתח חדש עם הלקוח דרך שרת הפרוייקט. את המפתח היא שומרת ב-DB
4. Get_new_client_with_key()
הפונקציה משמשת כדי לקבל את המפתח הישן של הלקוח דרך שרת הפרוייקט. את המפתח שהיא קיבלה היא ונרת במסד נותנים.
5. Forgot_password()
הפונקציה משמשת כדי לשלוח ללקוח את המפתח שלו.