**<שם הפרויקט>**

מסמך עיצוב

<עדי בלייאר>

<גרסה X>

<תאריך>

**היסטוריית גרסאות המסמך**

|  |  |  |
| --- | --- | --- |
| **תאריך** | גרסה | **תקציר השינויים** |
| 20.1.19 | 0.3 | פרק 2 ן 3 מעודכנים לתקופה |
| 5.4.19 | 0.4 | עדכון פרק 2 |
| 19.4.19 | 0.5 | עדכון כל הפרקים, גרסה 1 של סרבר בלבד. |
| 20.4.19 | 0.6 | הוספה של כל התרשימים המתוקנים. |



**1. הקדמה**

החלק הזה ישמש להצגה כללית של מסמך העיצוב. הסבר כללי, וקישור למסמכים קודמים כמו מסמך האפיון עליו אנו מתבססים

1.1 מטרה

*מה המטרה של המסמך ומי קהל היעד שלו*

1.2 המוצר

*חלק זה צריך להכיל את הדברים הבאים (ניתן לתמצת מתוך מסמך האפיון):*

* *שם המוצר אותו מפתחים*
* *הסבר כללי של מה המוצר עושה*

1.3 קישור למסמכים קודמים

*קישור למסמכים / קבצים חיצוניים הרלוונטיים למסמך. לדוגמא: מסמך אפיון, עמוד web עם עיצוב הממשק הכללי שאתם מתכננים וכו'.*

1.4 הגדרות

*רשומה – שם משתמש וסיסמא של שירות אחר.*

1. ארכיטקטורת המערכת

חלק זה כולל את תיאור מבנה המערכת ופירוט המודולים השונים בה

2.1 מבט על

המערכת מחולקת לצד שרת וצד לקוח.

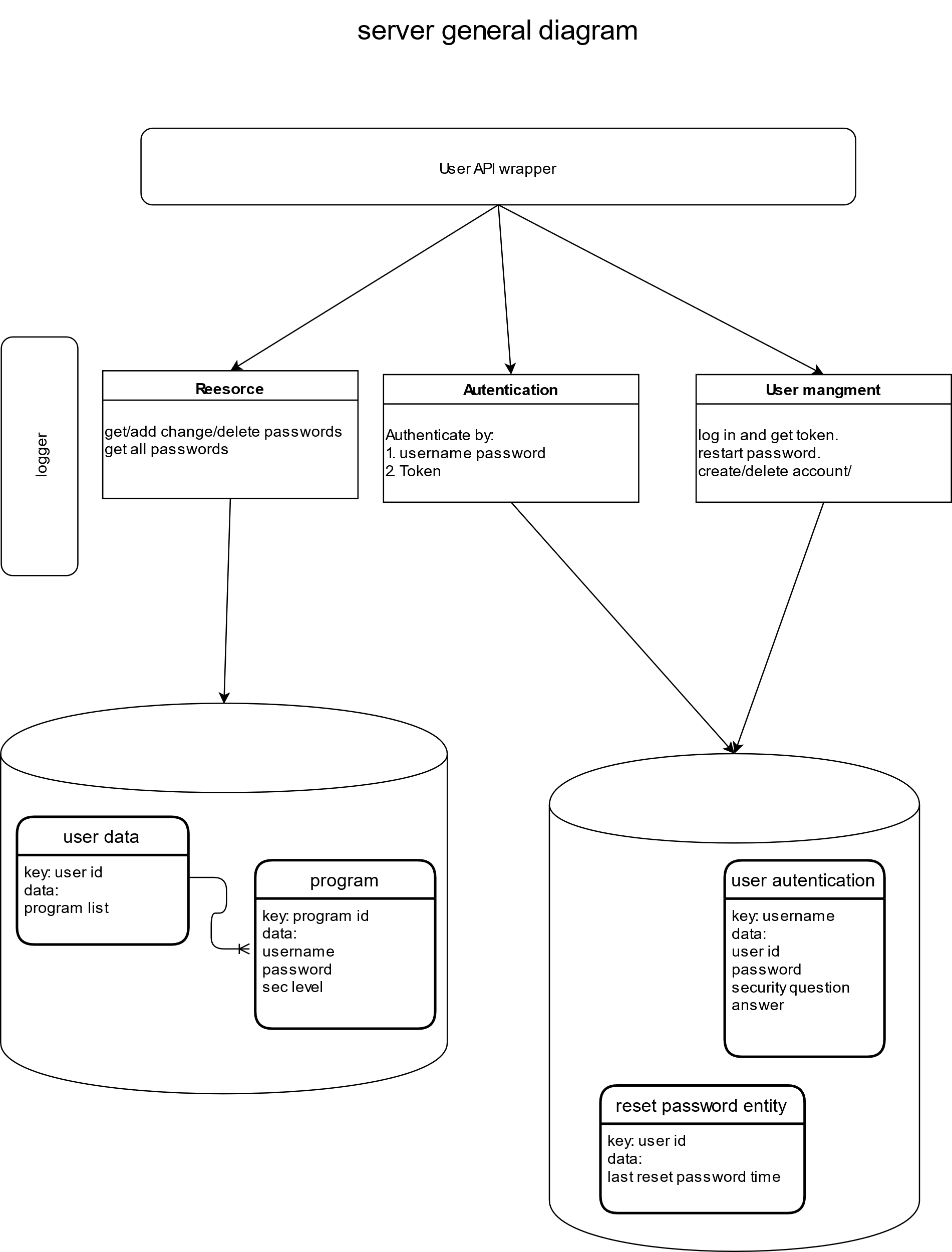
בצד שרת יש שלושה חלקים מרכזיים:

Resource- מודול המשאבים. מקבל בקשה, מוודא את תקינותה ומחזיר תשובה לפי ממסד הנתונים.

Authentication- מודול זיהוי משתמשים. מטפל בחלוקה ובדיקה של טוקנים ובהרשמה, שחזור ומחיקת חשבון.

User Management: ניהול המשתמש. הרשמה מחיקה ושחכתי סיסמא.

2.2 פירוט רכיבי המערכת

****

**רוב המודולים בנויים במבנה הבא:**

* Database **– כל הפעולות על ממסד הנתונים ששייכים לנושא זה**
* Request **– מכיל מחלקה\ מחלקות שמאחדות** URI **בעלי מאפיינים דומים. מכיל פונקציית ולידציה וקובע באיזה תרחיש ניתן לשהתמש ב**URI **אלה (רמת אבטחה, לפני או אחרי הזדהות)**
* API**- מכיל את מחלקות ה**URI**, שממשות את הפונקציונליות השונות לכל כתובות** URI.

authentication:

מכיל שני מודולים:

JWT: מטפל ביצור וולידציה של מפתחות.

מכיל ארבעה קבצים:

Create: יוצר מפתחות. קובץ פנימי.

Validate: מוודא את תקינות המפתחות הבקשה והעם היא מכילה מפתח שכזה.

Database: ממשק לבסיס הנתונים. בסיס נתונים בו ישמרו זמני שחזור הסיסמא האחרונים של המשתמשים, כדי שמפתח לא יהיה תקין במקרה ויוצר לפני שחזור\שינוי הסיסמא. – reset password entitiy

Request: מוסיף ולידציה של בקשת הזדהות עם טוקן.

Password: מטפל בהזדהות עם שם משתמש וסיסמא.

מכיל שלושה קבצים:

API- מיצוי הפונקציות החיצוניות של ממסד הנתונים, שינוי הודעות השגיא לקוד HTTP.

Database: פונקציונליות ממסד הנתונים ששומר שמות ושם משתמש וסיסמאות להזדהות. – user authentication

Request: מוסיף ולידציה של בקשת הזדהות עם שם משתמש וסיסמא.

Resource:

מכיל שלושה קבצים:

API- שני URL אחד לרשומה (קבלה, שינוי ומחיקה) ואחד למשתמש (קבלת כל הרשומות, הוספת רשומה, שינוי הגדרות חשבון, מחיקת חשבון.)

Database: פונקציונליות ממסד הנתונים ששומר את רשומות המשתמש.- user data + program

Request: מכיל מחלקה של הזדהות עם סיסמא, מאפשר ולידציה לנתונים.

User Management:

מכיל שלושה קבצים:

API- שלושה URL להתחברות, הרשמה ושחזור סיסמא.

Request: מכיל את המחלקות הנחוצות.

הזדהו על ידיד סיסמא בלבד (להתחברות).

בדיקת מבנה של הזדהות על פי סיסמא (להרשמה).

הזדהות על פי שם משתמש בלבד (לאיפוס סיסמא).

ישנם עוד מספר קבצים כלליים:

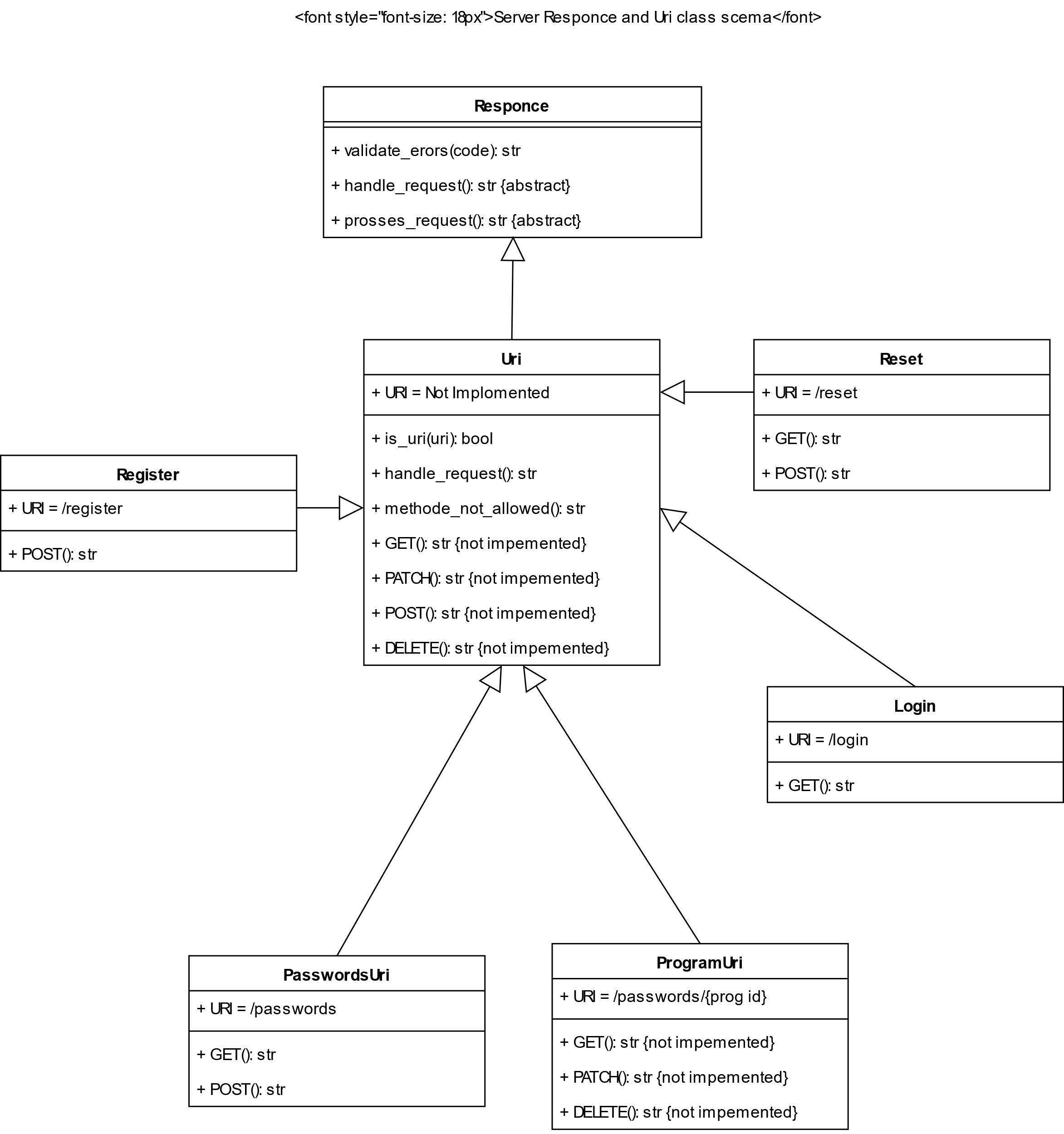
Request: מכיל מחלקה המאחדת בין שני סוגי ההזדהות ומספקת ממשק יחיד לכל שיטות ההזדהות. מאפשר קבלת user id ואת רמת ההבטחה של שיטת ההזדהות.

API: המסמך הראשי. מקבל בקשה מלקוח לקבלת מפתח ומחזיר הודעה חזרה (שגיאה או מפתח) או מקבל בקשה מלקוח לקבלת משאב ומוודא שללקוח יש הרשאה לכך.

HTTPtolls: מכיל מחלקת response המרכיבה תגובות http מהמידע בנוסף מחלקה URI אבסטרקטית המגדירה את הפעולות של מחלקות מסוג זה.

Database\_errors: מטפלת בכל שגיאות הפנייה לממסד הנתונים.

Request: מכיל את המחלקה reqest הבסיסית. מפלטרת בקשת http ונותנת יכולות בסיסיות של קריאת המידע.

****

**לכל סוג בקשות בחרתי לתת מחלקה כדי לאפשר חלוקה לסוגים, מאפשר לקבץ** URI **בעלי מאפיינים דומים.**

**חלוקת ה**URI **כדלקמן:**

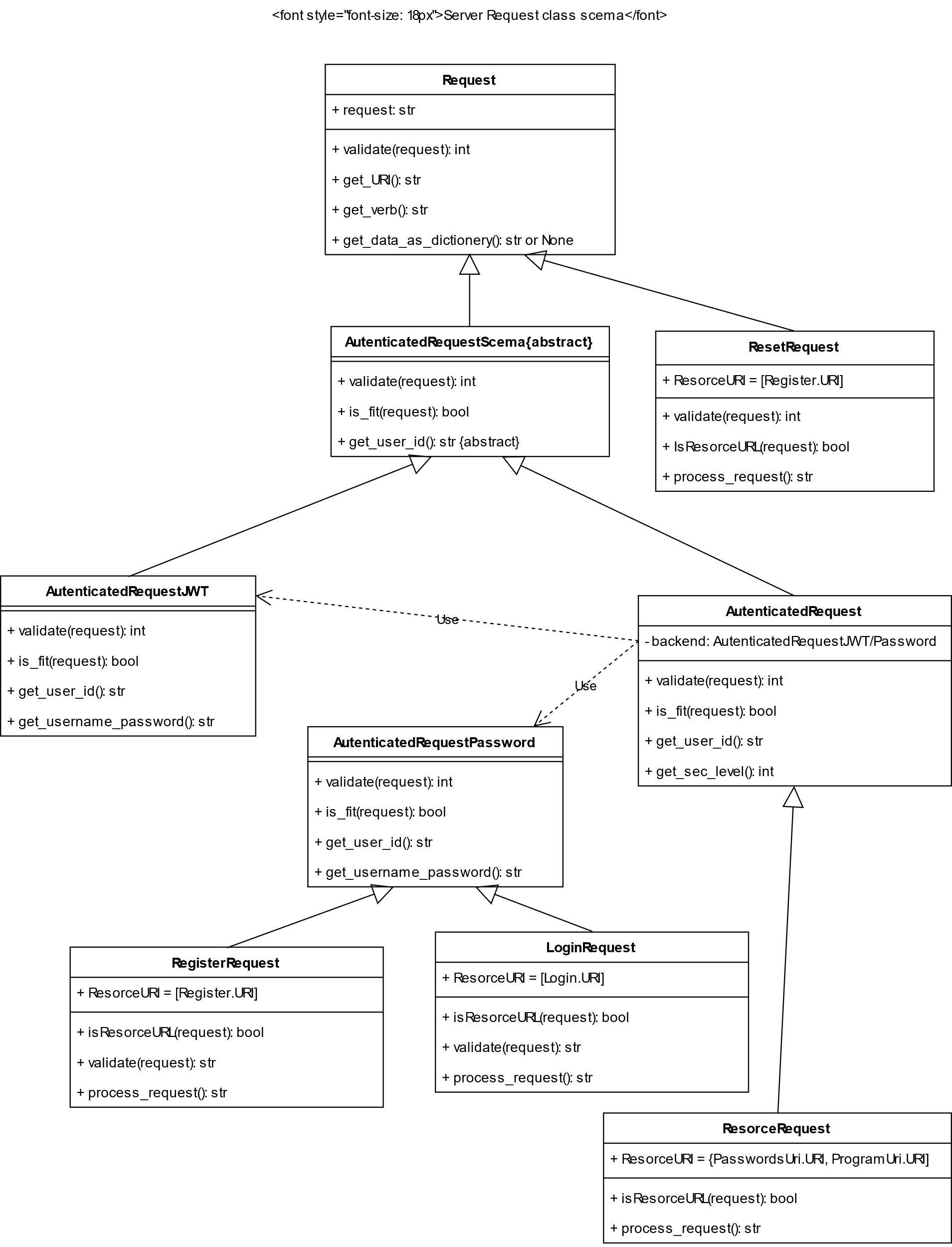
ReserRequest**- בקשות לפני הזדהות- שינוי סיסמא**

AutenticatedRequest**- בקשות לאחר הזדהות, לא משנה באיזו שיטה- בקשה להתחברות**

LoginRequest **– לאחר הזדהות על ידי סיסמא - התחברות**

RegisterRequest **– לאחר בקשה עם הזדהות תקינה אך לא קיימת במערכת – הרשמה**

**טעות בדיאגרמה AutenticateRequestPassword יורש מ Autenticated Request scema**

****

האם להוסיף לפני מסמכי התלבטות??

**בחרתי לחלק את השרת לשלושה רכיבים מרכזיים. כל רכיב לוקח אליו תפקיד אחד משמעותי. זיהוי המשתמשים, טיפול בבקשות קבלת המידע שלהם וטיפול בבקשות הגדרות המשתמש.**

**לכל סוג בקשות בחרתי לתת מחלקה כדי לאפשר חלוקה לסוגים, מאפשר לקבץ URI בעלי מאפיינים דומים. החלוקה לסוגים אפשרה לקבוע לאיזה פעולות למשתמש יש גישה אליהם לפי ההזדהות ששלח.**

**בחרתי בעיצוב זה כדי להפריד את ההזדהות וניהול ההרשאות של כל משתמש מהפעולות. מחלקות ה Request לסוגיהם מגדירות סוגים שונים של בקשות לפי ההזדהות הנדרשת בעוד שמחלקות הURI מטפלות במשתמש לאחר שהזדהה וידוע שיש לו הרשאות לפעולה זו (חוץ ממקרה אחד של בדיקת רמת ההבטחה בProgramUri כדי לאפשר רמות הבטחה שונות לרשומות שונות).**

1. עיצוב נתונים ופרוטוקולים

כאן יופיע תיעוד של מבני נתונים שונים / פרוטוקולים המשמשים אותנו במערכת

בסעיף זה יש לפרט את כל סוגי המידע אשר מועבר או נשמר במערכת - בין אם בין רכיבים, בין מודולים או בכל צורה אחרת (כלומר גם אם זה משמש רק לתקשורת פנימית ולא ע"ג הרשת, או למשל אם זה משהו ששומרים לקובץ/בסיס נתונים).

* לכל סוג מידע כזה יש לפרט על השדות אותן הוא מכיל ומאיזה סוג כל שדה, מה טווח הערכים הרלוונטי לגביו וכל הגבלה או מידע נוסף אחר שאתם מוצאים לנכון (למשל: האם מותר שהשדה יהיה ריק ומה המשמעות של זה, אורך או ערך מינימלי/מקסימלי למספר או למחרוזת)
* יש לציין לאיזו מטרה משמש כל מבנה נתונים.
* עבור פרוטוקול תקשורת-נתונים בין רכיבים שונים:
* מה המצבים השונים לכל סוג של תקשורת (למשל אותנטיקציה, הרשמה, שליחת קובץ וכו')
* באילו ייצוגי מידע נעשה שימוש + תרשים זרימה של המצבים השונים שיכולים להיות וסדר השלבים שלהם (למשל כיצד נעביר מידע על מקרים של סיסמה שגויה, משתמש חסום, התחברות מוצלחת וכו')
* מומלץ להגדיר מראש קודים של בקשות/פעולות, תוצאות/תגובות ושגיאות אשר ישותפו בין כך הרכיבים הרלוונטיים
* יש להתייחס לאופי המידע הנשמר (האם הוא טקסטואלי או בינארי) ולייצגו בצורה נוחה והולמת בהתאם
* יש לפרט אודות מבנה בסיס הנתונים (חלוקה לטבלאות, השם והטיפוס של כל עמודה/שדה, קשרים בין טבלאות ואילוצים כלשהם על עמודות)

לוגים: לוגים נשמרים בתיקיה logs. הלוגים מכילים את כל הלוגים מכל המרכיבים השונים של השרת. שמורים באופן טקסטואלי.

מפתחות הצפנה: נמצאים בקבצים בתיקיות הקוד.

הערה: מבנה הAPI הכללי של התוכנה מוגדר בקובץ נפרד API documentation

בסיסי נתונים - mongo dB שומר נתונים בצורת JSON וכך נכתב בסיס הנתונים.

1. זיהוי ואימות משתמשים.
   1. זיהוי משתמשים
      1. אינדקס ראשי: שם משתמש
      2. סיסמאת המשתמש (hash+salt)
   2. אימות טוקנים:
      1. אינדקס ראשי: מספר משתמש
      2. התאריך האחרון של שחזור סיסמא.
   3. מספר משתמש (אינו חשוף למשתמש וייחודי לכל משתמש.)
2. משאבי משתמשים
   1. מספר משתמש (אותו אחד כמו בממסד נתונים הקודם, ישמש לזיהוי המשתמש)
   2. האתרים והסיסמאות של המשתמשים – רשימה המכילה:
      1. מזהה ייחודי לתוכנה (הURL של אותו אתר ב safe\_url base 64)
      2. שם המשתמש לתוכנה
      3. סיסמא לתוכנה
      4. רמת אבטחה

4. ממשק משתמש

כאן יופיע פירוט ממשקי המשתמשים וכן תופיע הסקיצה עבורם

בסעיף זה יש לפרט את הפונקציונליות של המערכת כפי שהיא מתבטאת עבור משתמש חיצוני. במידה ויש סוגים שונים של משתמשים, יש להתייחס לכולם בהתאם - אילו רכיבים או נתונים רלוונטיים עבורם, וכיצד הם מתקשרים איתם.  
  
בנוסף, יש לצרף המחשות ויזואליות של המסכים השונים, ולהסביר את התוכן שלהם (למשל מה התפקיד של כל שדה או כפתור, מה קשור/תלוי במה - למשל כפתור שמכובה בהתאם לתנאים מסוימים במערכת וכן הלאה) והקשרים ביניהם (איזה מסך מוביל לאיזה מסך ובאילו מקרים)  
  
דוגמה (לא מלאה):  


5. נספחים

**כל דבר שהייתם רוצים להוסיף בנוגע לעיצוב המערכת**