**<שם הפרויקט>**

מסמך עיצוב

<שם הכותב>

<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 |
|  |  |  |
|  |  |  |



**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

*בחלק זה יש לציין את כל ההגדרות, המושגים וראשי התיבות בהם תשתמשו בהמשך המסמך.*

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 שני חלקים מרכזיים:

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

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

קבצים נוספים חשובים:

Logs: קובץ שמתפל בלוגים. מאפשר קבלת לוגר על ידי התוכנות השונות ותומך בשינוי ההגדרות שלו בזמן ריצה.

2.2 פירוט רכיבי המערכת

**כאן יופיע פירוט לכל רכיב (ניתן לעשות באיטרציות – כל פעם עבור הרכיבים הרלוונטיים, לפני תהליך הפיתוח שלהם)**

**רוב המודולים בנויים במבנה הבא:**

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

צד שרת-resource:

מחולק לשלושה קבצים שונים.

reqest: מכיל את מחלקת ה Request שמכילה את כל הURI

Database: מכיל את כל הפונקיות שפונות לממסד הנתונים.

API: המסמך הראשי. משלב את הכל למחלקות URI. הפונקציה היחידה שחשופה מבחוץ. מקבלת בקשה ומחזירה את המידע לשלוח חזרה. שגיאה או המידע השתבש.

צד שרת-authentication:

מכיל מודול אחד:

JWT: מטפל ביצור וולידציה של מפתחות.

מכיל שלושה 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 הסיסימא.

Password: מטפל בהזדהות עם שם משתמש וסיסמא.

מכיל שלושה קבצים:

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

Database: פונקציונילות ממסד הנתונים ששומר שמות ושתמש וסיסמאות להזדאות.

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

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

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 מהמידעץ בנוס מחלקה URI אבסטרקטית המגדירה את הפעולות של מחלקות מסוג זה.

Database\_errors: מתפלת בכל שגיאות הפנייה לממסד הנתונים.

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

בסעיף זה יש לפרט את המבנה הפנימי של כל מודול/רכיב - כלומר:

* מבנה המחלקות השונות המרכיבות אותו, המאפיינים שלהם (public/private/protected + אילו טיפוסי משתנים כל פונקציה מקבלת ו/או מחזירה) והקשרים ביניהם (מי קורא למי, משתמש במי ולאיזה צורך) – מומלץ לצייר גרף UML (לא חייבים לפרט משתנים/פונקציות פרטיות אבל שיהיה ברור מה קשור למה ובאיזה אופן או לאיזה צורך)
* יש לתאר את ה- data flow / use case של שימושים שונים בכל תת-רכיב או פונקציה (מה קורה אם מתקבלים סוגים שונים של ערכים, איך/מדוע/באילו מקרים הרכיבים מתנהגים בצורה מסוימת או אחרת וכן הלאה)

**המלצות:**

* כדאי לתכנן רכיבים תשתיתיים עבור צרכים גלובליים במערכת - כמו למשל יצירת תקשורת רשתית, כתיבה וקריאה לבסיס נתונים (או קבצים), כתיבת תיעוד (log) הפעולות של המערכת (מקובל להגדיר "רמה" לכל שורה שנכתבת - debug, info, warning, error כדי שיהיה קל לסנן בזמן ריצת התוכנית. בנושא זה – מוזמנים להסתכל על מודול logging בפייתון).
* מומלץ לתכנן את הרכיבים באופן שיאפשר אבסטרקציה ומודולריות – בכדי לאפשר הרחבה עתידית ותחזוקה קלה. אם המימוש הפנימי של רכיב או מודול מסוים משתנה אז שאר הרכיבים/מודולים אינם מושפעים או נפגעים מכך ואין צורך לשנות גם אותם, או כאשר רוצים להוסיף פונקציונליות/רכיב חדש אז אין צורך לשכתב את כל שאר המערכת במיוחד בשבילו.

2.3 דיון בנושא העיצוב הנבחר

**כאן יש לפרט ובעיקר להסביר מדוע בחרתם דווקא בחלוקת הרכיבים/תפקידים הזו ולא אחרת. מה היתרונות שלכם מבחינתה, וגם מה החסרונות שאתם מודעים אליהם. במידה וחשבתם על חלופות אחרות, יש לציין אותן ולהסביר בקצרה כיצד הן שונות מהפתרון שנבחר - ומדוע החלטתם בסופו של דבר שלא לבחור בחלופות האלו.  
כאן יש להתייחס גם לגבי שפת התכנות שבה בחרתם לכתוב כל רכיב, ומדוע דווקא בשפה זו ולא אחרת.**

**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.**

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: איך לשמור אותם???

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

בסיסי נתונים:

1. זיהוי ואימות משתמשים.
   1. אמצעי זיהוי (שם משתמש וסיסמא עם ריפוד וhash).
   2. מספר משתמש (אינו חשוף למשתמש וייחודי לכל משתמש.)
   3. מידע לאימות דו שלבי.
   4. מספר טלפון\חשבון נאמן לשחזור חשבון.
   5. הגדרות האבטחה של המשתמש
2. משאבי משתמשים
   1. מספר משתמש (אותו אחד כמו בממסד נתונים הקודם, ישמש לזיהוי המשתמש)
   2. האתרים והסיסמאות של המשתמשים.

בנוי כJSON בפורמט להלן:

{

"\_id":{"$oid":"aaaaaaaaaaaaaaaaaaaaaaaa"},

"records":[

{"username":"username1","password":" password1","program\_id":"program1"},

{"username":" usernameN","password":" passwordN","program\_id":" programN"}

]

}

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**