beek**DB**

מסמך אפיון

יותם ברק תיכון הדרים

**היסטוריית גרסאות המסמך**

|  |  |  |
| --- | --- | --- |
| **תאריך** | גרסה | **תקציר השינויים** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |



**1. הקדמה**

החלק הזה ישמש להצגה כללית של מסמך האפיון - הוא נועד לאפשר לקורא להבין במה מדובר - מה זה בכלל הפרויקט הזה ומה מופיע בהמשך המסמך.

כשרציתי לבחור פרוייקט היו לי התלבטויות רבות גם על פרוייקטים לעשות, וגם על מה יהיה בהם. לבסוף הגעתי להחלטה סופית ואני גאה במה שיצרתי. המסמך לפניכם יסביר את הפרוייקט לעומקו, ויאפשר הבנה טובה שלו, של המהות שלו, ושל איך הוא פועל ועובד.

1.1 מטרה

*מה המטרה של המסמך ומי קהל היעד שלו*

המטרת המסמך היא שלאחר שתקראו אותו תוכלו להבין את הפרוייקט על בוריו ולהעריך את העבודה כראוי.

אני מקווה שאחרי שתקראו את הספר תוכלו להבין טוב יותר את הפרוייקט ולהסתדר עם הקוד.

1.2 המוצר

*חלק זה צריך להכיל את הדברים הבאים:*

* *שם המוצר אותו מפתחים (ובאמצעותו נתייחס למוצר בהמשך המסמך)*
* *הסבר כללי של מה המוצר עושה (ואם צריך- מה הוא לא עושה)*
* *תיאור של מטרות המוצר ויתרונותיו העיקריים. נסו להיות מדויקים וספציפיים כמה שניתן.*

*הפרוייקט שעשיתי הוא בסיס נתונים בשם beekDB. הוא בעצם דאטהבייס ותוכנה שבעזרתה מתקשרים איתו. הוא משמש לשמירת כמות גדולה של נתונים בצורה יעילה והתוכנה מממשת שרת מרובה לקוחות ומאפשרת אבסטרקציה לפעולות על בסיס הנתונים כמו קריאה, כתיבה, עדכון וכולי. התוכנה תתמוך בממשק באמצעות שפת SQL.*

*המטרה של שימוש ב beekDB היא הפרדה לוגית בין שמירת הנתונים לבין התוכנה עצמה (לדוגמה במערכת כמו אמזון ששמירת נתונים כמו לדוגמה הזמנות וחשבונות ייטופלו על ידי בסיס הנתונים ככה שהתוכנה עצמה לא תתעסק בדברים כמו כתיבת הנתונים לדיסק הקשיח.*

*התוכנה תרוץ על המחשב ואליה יהיה ניתן להתחבר דרך ה API שאני אממש בשפות שונות.*

1.3 הגדרות

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

*בספר הזה אשתמש במונחים להלן:*

* *הסרבר (תוכנת בסיס הנתונים אליה יהיה אפשר להתחבר בתור לקוחות. הוא אחראי על תקשורת עם הלקוחות ושמירת המידע בדיסק)*
* *קליאנט (לקוח שמתממשק עם ה API של הסרבר)*
* *פרוטוקול תקשורת (הפרוטוקול איתו קליאנטים יתקשרו עם הסרבר (אולי דרך SQL)*
* *פרוטוקול דיסק (הפרוטוקול בו הסרבר כותב לדיסק דברים בצורה יעילה)*
* *ממשק משתמש (לסרבר יהיה ממשק משתמש גרפי)*
* *מודול תקשורת (מודול שאני אכתוב ב++C שיהיה wrapper לAPI של הסוקט בלינוקס)*
* *Result Set הכוונה הפלט של שאילתת SQL שהיא בעצם טבלה עם התוצאה שהתקבלה.*

**1.4** תקציר

*הסבר מה מופיע בהמשך המסמך ופירוט כיצד זה מסודר לאורך המסמך*

*במסמך זה, יפורט הרבה על הפרוייקט ועל מה יש בו, כמו גם על הדרישות שיש לי שיהיו בו.*

1. תיאור כללי

החלק הזה מתאר את הגורמים העיקריים המשפיעים על המוצר ועל דרישותיו. חלק הזה לא מפרט דרישות ספציפיות אלא רק עוזר להבין את הדרישות.

*הפרוייקט בעצם מורכב מסרבר שהוא תוכנה שרצה ומאחסנת דאטה בזיכרון על פי ה QUERYS שהיא מקבלת מהלקוחות, שהוא יכיל גם ממשק משתמש גרפי שיאפשר לבצע כל מיני פעולות בבסיס הנתונים, וגם הפרוייקט יכיל מימושים ל API של הסרבר בכל מיני שפות דוגמת ג’אווה, פייתון וכולי.*

*אני שואף שה QUERYS יהיו בשפת SQL, שתיתמך בתוכנה בחלקה, מסיבה זו התוכנה תצטרך לעשות Text Parsing לקוד ב SQL.*

2.1 פונקציונליות

*חלק זה יכיל סיכום של הפונקציות העיקריות במערכת (לא פונקציות של שפת תכנות, אלא באופן מילולי - מה הדברים שהמערכת עושה)*

*אלגוריתמים של כתיבה לדיסק:*

*כיצד הסרבר כותב דאטה לדיסק בצורה יעילה, שניתנת לשליפה ולהכנסה בקלות, בלי צורך בהרבה חישוביות או כתיבה לזיכרון (לדוגמה להמנע מלהזיז הרבה קבצים בשביל הכנסה)*

*תקשורת מרובה לקוחות:*

*להיות מסוגל עם סוקטים ותהליכונים לתקשר עם מספר לקוחות במקביל, ולהמנע מקונפליקטים בין התהליכונים דוגמת DEADLOCK.*

*ניתוח טקסטואלי של שפת SQL (לפחות החלק המהותי)*

*יהיו טרנזקציות שיממשו את עקרונות ACID.*

2.2 קהל היעד

*תיאור כללי של קהל היעד של המערכת - מה תפקידם, ניסיונם, יכולתם ללמוד את ממשק המערכת וכל דבר אחר שישפיע על דרישות המערכת. יתכן יותר מפרופיל אחד של משתמש במערכת, ולכן יש לפרט את כל הסוגים וההשלכות/משמעויות שלהם מבחינת אופי ורקע טכנולוגי או פונקציונלי.* ("מי הולך להשתמש במערכת?")

*קהל היעד של המערכת יהיה אנשים שמתכנתים במגוון שפות תכנות, וצריכים דרך יעילה לאחסן נתונים בדיסק, במקום לכתוב בקבצים לבד בדרכים לא יעילות, אפשר להשתמש ב BeekDB.*

*הם צריכים ניסיון בשפת SQL על מנת לתקשר עם בסיס הנתונים, וניסיון כמובן בשפה שאיתה הם מתקשרים.*

*בשביל אנשים יותר פרימיטיביים, יהיה גם לסרבר ממשק משתמש שבו יהיה אפשר לבצע פעולות סטנדרטיות על בסיס הנתונים.*

2.3 אילוצים עיקריים

*הגבלות ואילוצים עיקריים שצריך לקחת בחשבון בזמן עיצוב המערכת - אילוצי חומרה, סביבת ריצה, זמן ריצה וכו'..* (למשל – "המערכת דורשת חיבור לאינטרנט" / "המערכת צריכה לתמוך במספר רב של מערכות הפעלה" / "המערכת צריכה לתמוך במספר משתמשים רב")

*אילוצים קריטיים של המערכת יהיו:*

*אני אנסה לממש כמה שיותר משפת SQL, אבל ייתכן מאוד שלא אממש אותה בשלמותה מכיוון שהיא שפה עם הרבה מאוד דברים.*

*המערכת לא דורשת חיבור לאינטרנט כלל, מכיוון שהדאטהבייס רץ מקומית על ה LOCALHOST.*

2.4 הנחות ותלויות

*כל הנחה שיש לנו לגבי המערכת וסביבת הריצה שלה, ששינוי שלה עשוי לגרום לשינוי באפיון ובעיצוב - לדוגמא "המערכת תרוץ בסביבת לינוקס" זו הנחה המשפיעה על אפיון ועיצוב המערכת. אם נשנה את זה בהמשך נצטרך לעבור שוב על המסמך ולשנות את אפיון ועיצוב המערכת בהתאם (זה שונה מהסעיף הקודם- אילוצים זה לא משהו שיכול להשתנות, זה משהו שנקבע בתחילת העבודה ומשפיע רק על האפיון הראשוני)*. אל תתנו להנחות הללו להגביל אתכם בהמשך (אם תגלו שלמשל עדיף בסוף להריץ את השרת המרכזי מעל וינדוס – עשו זאת).

**2.5** סיקור מצב השוק כיום (אופציונלי)

*במידה וידוע לכם על מוצרים קיימים כיום (אתרים, תוכנות, אפליקציות וכו') שנותנים פתרון דומה (גם אם חלקי) לרעיון שלכם - קשרו אליהם כאן או תארו אותם בקצרה. התמקדו באופן שבו הם פותרים או מתייחסים לסוגיות דומות לאלו שבחרתם, מה הפיצ'רים העיקריים שהם מציעים ובמה הם שונים מהמוצר שלכם.*

1. דרישות מפורטות

החלק העיקרי והחשוב ביותר של מסמך האפיון. פרק זה יכיל את הדרישות המפורטות מהמערכת שינחו את מעצבי המערכת, המפתחים, והבודקים בהמשך. כל דרישה שתפורט בחלק זה צריכה להיות:

* מזוהה על ידי שם (או עדיף - מספר) באופן ייחודי
* מתועדפת (כלומר מה העדיפות שלה כלפי הדרישות האחרות)
* ניתנת לבדיקה
* נכונה ושלמה
* עקבית ולא מתנגשת עם דרישות אחרות

*שימו לב שישנם מספר סוגים של דרישות, כפי שמפורט בכותרות כאן:*

*יש להשקיע בחלק זה ולפרט את הדרישות בצורה ברורה ונוחה להבנה. יש לזכור שזה לא מסמך עיצוב- יש להמנע מתיאור מפורט מדי שמגביל את העיצוב הטכני (למשל – שפת תכנות) שיגיע בשלב מאוחר יותר של עבודתכם.*

*את הדרישות רצוי לחלק לקטגוריות הבאות:*

3.1 דרישות פונקציונליות

*תיאור של פיצ'רים ספציפיים במערכת. לכל דרישה פונקציונלית מומלץ לפרט:*

* *הסבר כללי*
* *קלט*
* *תהליך*
* *פלט*
* *טיפול בשגיאות*

*ניתן (ואף רצוי) להציג את הדרישות הפונקציונליות בפורמט של* [*תרחיש שימוש*](http://he.wikipedia.org/wiki/תרחיש_שימוש) *(*[*use case)*](http://he.wikipedia.org/wiki/תרחיש_שימוש)*. בכל "נקודת זמן" של תיאור התרחיש יש לפרט את כל האפשרויות השונות שעומדות בפני המשתמש או המערכת, ומה יכול לקרות בכל התפצלות שכזו (שימו לב להתייחס גם למקרים שליליים, כמו שגיאות או נתונים לא מתאימים, ולא להניח שהמשתמש תמיד פועל בדרך שלה אנו מצפים)*

אבסטרקציה להתממשקות עם סוקט רגיל בלינוקס ויצירת קלאס סוקט על מנת לעבוד איתו בקלות במימוש הפרוטוקול. יכיל פעולות כמו RECV, SEND, SOCKET, BIND, וכו. שבעצם יהוו מעטפת לאלה הבנויות במערכת ההפעלה.

פרוטוקול תקשורת בין כלשהו לקוח של בסיס הנתונים לשרת שייתן פונקציות פשוטות של SEND, RECV. בעזרתו יעברו כל ההודעות על הרשת.

מימוש שרת מרובה לקוחות שאיתו אפשר יהיה לתקשר עם מספר לקוחות במקביל.

פירוש בסיסי של QUERIES בשפת SQL. יידע לבצע על בסיס הנתונים שאילתות SQL בסיסיות לפחות.

בטבלה יהיו מספר טיפוסים אפשריים, ביניהם מספר שלם (INTEGER), מספר ממשי (REAL), טקסט כלשהו עם אורך מוגבל למספר תווים מסוים (VARCHAR) על מנת לאפשר כתיבה וקריאה מהירה.

התקשורת עם הסרבר תהיה כך:

הלקוח ישלח שאילתת SQL לשרת, שיגיב בתגובה בטבלה כלשהי, בהתאם. הטבלה תישלח דרך הפרוטוקול חזרה ובעצם כל מימוש של לקוח תמיר אותה למבנה הנתונים שהגיוני לשפה. (לדוגמה ל DICTIONARY של רשימות בפייתון).

3.2 דרישות של ממשקים חיצוניים

**3.2.1 ממשקי משתמש** (למשל: GUI)

**3.2.2 ממשקי חומרה**

**3.2.3 ממשקי תוכנה** (אם התכנה מחולקת לכמה תוכנות שונות שצריכות לתקשר בניהן)

**3.2.4 ממשקי תקשורת (למשל תקשורת בין שר**ת ללקוחות)

כרגע ההתממשקות עם הסוקט היא תלויית מערכת הפעלה, כלומר היא משתמשת בסוקטים בסטנדרט של POSIX, שווינדוס לא מממש. אני גם אוסיף מימוש דומה לווינדוס ככה ששימוש בווינדוס יהיה אפשרי.

3.3 דרישות לא פונקציונליות

*דרישות מהמערכת שלא מתבטאות בפיצ'ר ספציפי או בתהליך ספציפי שמתרחש במערכת אבל משפיעות על אופן עיצובה ומימושה, לדוגמא:*

**3.3.1 דרישות ביצועים (performance)**

**3.3.2 דרישות מהימנות (reliability)**

**3.3.3 דרישות זמינות (Availability)**

**3.3.4 דרישות אבטחה (security)**

**3.3.5 דרישות תחזוקה (maintainability)**

**3.3.6 דרישות ניידות (portability)**

בסיס הנתונים צריך להיות מהימן ולשמור על לפחות חלק מעקרונות ACID.

נדרשת גם הצפנה של נתונים יותר רגישים ככה שצד שלישי לא יוכל להסניף אותן על הרשת.

3.5 דרישות נוספות

*כאלה שלא ידעתם לאיזה חלק הם שייכות*

5. נספחים