**פרויקט שנתי – תיאור**

**תוכן עניינים**

[כללי](#כללי)

[ארכיטקטורה](#ארכיטקטורה)

[תהליך הפיתוח](#תהליך)

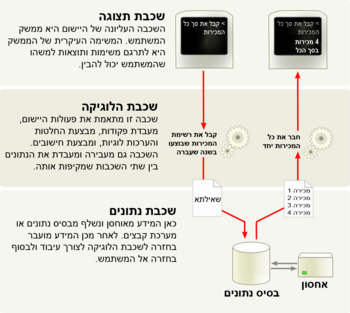
**כללי**

הפרויקט נוצר על מנת לתמוך במחסן עצים, בו לכל מוצר אורך קבוע (בנוסף לתכונות נוספות), אשר צריך להתאים אורכים משתנים עפ"י דרישות לקוחות שונים כך שחיתוך האורך הקבוע של כל מוצר ישאיר פחת מינימלי.

הפרויקט מבוסס על טכנולוגיות WEB כלומר מודל לקוח-שרת מבוזר, אשר תומך בבקשות-תשובות מעל פרוטוקול HTTP או, יותר מאובטח HTTPS. התקשורת לקוח-שרת נעשית בצורה אסינכרונית משמע מרגע שבקשה נשלחת מהלקוח, אין כל הגבלה על המשך השימוש ו/או שליחת בקשות אחרות, טיפול בתשובה מהשרת נעשה ללא תלות במצב הלקוח.

**ארכיטקטורה**

הארכיטקטורה של המערכת מבוססת על מודל 3 השכבות, כפי שמתואר בויקיפדיה:



מעקב אחרי החיצים האדומים מתאר את תעבורת המידע מהלקוח שנצא אי שם ברשת, שולח בקשה לשרת, הבקשה מעובדת בשכבת הלוגיקה ומועברת לבסיס הנתונים, שיושב בשכבת הנתונים, ומשם התשובה נשלחת חזרה עם הנתונים אל הלקוח. שתי השכבות ה"תחתונות",שכבת הלוגיקה והנתונים, נמצאות על השרת ואילו העליונה על המכונה של הלקוח (דפדדפן).

**תהליך פיתוח:**

אסטרטגיית הפיתוח שבה נעשה שימוש היתה ניצול של כלי לניטור גרסאות, כלי בניית קוד ופלטפורמת פיתוח. הרעיון המוביל היה שימוש בכלים שפותחו ב"קוד פתוח" ולא תלויים במערכת הפעלה כזאת או אחרת, כך למשל, אחד מאיתנו פיתח במערכת Linux והשני השתמש ב Windows והמוצר הסופי לא הושפע בכלל, השינויים הוטמאו בפרויקט ללא קשר לצד שהכניס את השינויים.

תחת אסטרטגייה זאת, כל חבר בצוות כתב קוד מעל Eclipse שהיא סביבת פיתוח, קוד פתוח, המתאימה למגוון מערכות הפעלה, את הקוד מקור בלבד (!) כל חבר העלה לכלי ניטור גרסאות, במקרה זה github פלטפורמה פופולרית מאוד, את השינויים שכתב. מה שחסר הוא צעד ביניים בין קוד מקור לפרויקט בסביבת פיתוח, שלב זה נקרא כלי בנייה או build tool, במקרה הזה נעשה שימוש ב Maven, שהוא חלק מה Apache Software Foundation שבונה את סביבת הפיתוח המתאימה עם כל התלויות הנלוות, ספריות ומודולים חיצוניים.

כל אימת שהצוות ראה לנכון להוסיף עוד פונקציונאליות, החבר בצוות שקיבל את האחראיות הוסיף ב github ענף נפרד עם שם מתאים לצורך פיתוח הקוד ורק לאחר שנבדקה הפונקציונאליות ונמצאה מניחה את הדעת נעשה מיזוג של הקוד החדש עם הקוד הקיים בענף הראשי. לעיתים, חבר אחד בצוות עבד על שינויים בענף הראשי ואילו החבר הנוסף המשיך בעבודה על ענף נפרד, ולכן המיזוג הצריך פגישה על מנת לתאם שהשינויים לא מבטלים או מפריעים אחד לשני.

**מימוש**:

בעיה מרכזית

הפרויקט כאמור, מאפשר למעשה למחסן עצים לספק את דרישות הלקוחות תוך מיצוי המלאי בצורה מיטבית. משמע, חלוקה של מוצרים (קורות עץ בפרופילים שונים ואורך קבוע לכל פרופיל) לאורכים שונים לפי צבר הזמנות נתון. זוהי למעשה בעיית אופטימיזציה שאפשר לנסח באופן הבא:

בהינתן מוצר בעל אורך L ו k קטעים באורכים שונים, צריך לבחור קבוצה b מתוך B של קטעים כך שההפרש בין L לסכום אורכי הקטעים מהקבוצה b יהיה מינימלי וחיובי (סכום הקטעים לא יכול להיות גדול מ L). ובצורה יותר פורמאלית:

בהינתן אורך L וקבוצה B בגודל N של קטעים באורכים יש למצוא תת קבוצה b בגודל n כך שמתקיים

קל להשתכנע שהבעיה שייכת למשפחת NP שלמה.

קוד וסכימת הפתרון:

הפתרון המוצע כאן לבעיה המרכזית הוא פתרון Brute Force כלומר, חישוב כל האפשרויות השונות וזאת מכיוון שלא ידוע פתרון אחר, ובנוסף, גודל מרחב ההזמנות (הקבוצה B ) הצפוי בכל רגע נתון יחסית קטן לכן, כך שאפילו אם מדובר בכ 30 הזמנות, יש לבצע לא יותר מ חישובים.

את המערכת הוחלט לממש מעל הרשת במודל לקוח-שרת. המודל מאפשר הן גישה נוחה מכל מקום בעולם, והן מימוש מודל 3 השכבות (3 tier model) בצורה טבעית.

מימוש המערכת (כתיבת קוד) התבצע ע"י שימוש בפרויקט Apache Software Foundation שהוא פרויקט "קוד פתוח" המשלב מעל ל 40000 משתתפים מאז 1999. הסיבה לשימוש ב Apache היא הפופולריות העצומה של הפרויקט בקרב משתמשים רבים בעולם, כמו כן השימוש בשרת Tomcat ואיתו מסד הנתונים Derby בעל חתימה נמוכה (low foot print).

הפרויקט מתממשק היטב עם שפת התכנות **Java** שהיא שפת תכנות מוכוונת אובייקטים (object oriented) אבל, שלא כמו c++, היא שפה שבה הזכרון מנוהל ע"י הפלטפורמה שמפעילה אותה (managed code). פלטפורמה זאת נקראת JVM שלמעשה אלה ראשי התיבות של Java Virtual Machine שבניגוד לשפת תכנות OO אחרת כמו c# של Microsoft או objective c של Apple לא רצה על מערכת ההפעלה אלא על מכונה וירטואלית.

שרת Tomcat: תוכנה שלמעשה אחראית על כל תעבורת המידע במודל ה OSI, מקבלת בקשות HTTP ועד מציאת המכונה הנכונה ברשת. Tomcat היא הלכה למעשה השרת שאליו פונים וממנו מקבלים תשובות.

שרת Derby: תוכנה שמנהלת את המסד נתונים, מריצה שאילתות ואחראית לתקינות ושלמות הנתונים. Derby הוא למעשה שרת נתונים, משמע שרת המקבל בקשות לשאילתות מעל SQL שהיא שפת "שאילתות מובנית". מחינת השימוש בשרתי SQL הולך ודועך כאשר נפח התעבורה גדל משמעותית, אבל, לצרכי הפרויקט הוא מספק. קיימות שתי דרכים להפעיל את Derby:

**Embedded Mode:** מאפשר מצב של "רק קריאה" מהשרת, לא מאפשר המשך עיבוד נתונים עם שגיאה ומאפשר גישה לשרת גם ללא אימות שם משתמש+סיסמא (אלא רק שם משתמש).

**Network Client Mode**: ההיפך ממצב Embedded ביחס לנ"ל, מעבר לזאת, מצב זה מאפשר השמה אוטומטית של ערכי NULL כאשר קיימת המרה בין טיפוסים.

הבחירה במצב Embedded בפרויקט נבעה בעיקרה מדעה רווחת באתרי מפתחים בהם קיימת העדפה מסויימת למצב זה, בשלב שבו אנחנו נמצאים בתור מפתחים מתחילים נמצאים לא ראינו ממש הבדלים.

הטבלה הבאה מסכמת את השימוש בתוכנות והשפות בפרויקט:

|  |  |  |
| --- | --- | --- |
| שכבה | גירסה | שפה/תוכנה |
| שרת | 1.8.0 271 | Java Runtime Environment |
| 15.0.1 | Java Development Kit |
| 9.0.41 | Tomcat |
| 10.15.2.0 | Derby |
| מפתח | JEE 2021-06 (4.20.0) | Eclipse |
| 3.6.3 | Maven |
| לקוח | 5.0 | HTML |
| 2.1 | Cascade Style Sheet |
| ES6 ECMAScript 2018 | JavaScript |
| 5.0 | Bootstrap |
| 3.6 | JQuery |

**דרישות**

את מפרט הדרישות המלא של הפרויקט אפשר למצוא במסמך "דרישות פרויקט" (בשפה האנגלית). עם זאת, הדרישות העיקריות הן:

* אפשרות ניהול מלאי מוצרים למשתמש על (אדמיניסטרטור)
* יכולת גישה למערכת מכל מקום מעל הרשת
* רישום למערכת
* הזמנה של מוצרים (צבע, ואורך) עם/ללא רישום
* אפשרות שליחת הודעות בין כל המשתמשים הרשומים

**בדיקות**

את מפרט הבדיקות ניתן למצוא במסמך "מפרט דרישות" (בשפה האנגלית)

**קשיים**

הקשיים שבהם נתקלנו מתחלקים בעיקרון לשתי קבוצות:

הקמה: קבוצה זאת מכילה את חבלי הלידה של המערכת, איזו שפת פיתוח, ואיתה, איזה סביבת פיתוח. חשיבה על תהליך הפיתוח עצמו כפי שהוצג לעיל, איך מפתחים כך שלא תהייה תלות במערכת ההפעלה וכו'. הקשיים המאפיינים את הקבוצה הזאת בעיקר כללו החלטות עם השלכות על *תהליך הפיתוח*, ומכיוון שנסיון הצוות מוגבל, עיקר הפתרונות נבעו מסקירה של נסיון מפתחים קודמים כפי שהוצג באתרי מפתחים שונים מרחבי הרשת.

פיתוח: קבוצה זאת של קשיים מכילה את הלמידה תו"כ תנועה של שפות חדשות

* **Javascript**, היא שפה המיועדת עבור דפדפנים, תפקידה לספק פונקציונאליות משלימה לתצוגת האלאמנטים של HTML. היו פערי ידע בדרך בה הדפדפן מרנדר את המידע שכתוב בקבצי ה HTML וה CSS לכדי ממשק משתמש, סדר קריאה לפונקציות (כמו קריאה לאירועים-events שונים ויירוטם). כמו כן פערי ידע המכסים תעבורה של מידע מעל הרשת (פרוטוקולים וטכנולוגיות יישום שונות, כפי שמפורט להלן).
* **CSS**, cascade style sheet. הדפדפן מרנדר את האלמנטים של ה HTML בהתאם לתכונות שלהם, אחת מהתכונות הללו היא הסגנון, style, של האלמנטים. דרך תכונה זאת ניתן לקבוע מיקום, צבע, צורה,גבולות ועוד הרבה מאוד פרמטרים וכולם ביחד קובעים איך יוצג האלמנט. CSS היא דרך שבה ניתן להפריד בין מסמך ה HTML שאומר *אילו* אלמנטים ייראו ובין המסמך שקובע *איך* האלמנטים ייראו. הקושי היה להבין מהם הפרמטרים השונים ואיך הם משלימים או מבטלים אחד את השני. כמו כן, להבין מהו סדר הקסקדה, מכיוון שניתן להגדיר פרמטרים אלה דרך קבצי CSS. קבצי JS. ודרך קבצי HTML. וסדר ההגדרות משנה את תוצאת הרנדור והבנת כל הנ"ל הצריך לימוד.
* בעיה נוספת היתה ב**העברת תמונות** ושמירתן במסד הנתונים, והאם בכלל זה אפשרי, או האם ניתן לשמור על הfile system של המכונה שעליה מותקן השרת. לדעתנו האפשרות השנייה היתה סיכון מסויים שכן הכוונה היא שאת המערכת ניתן יהיה לפרוס (deploy) על כל מערכת הפעלה ולכן קיימת אפשרות ששמירה כזאת לא תתאפשר. מכאן, שמירה של תמונות (וקבצים בכלל) היתה אתגר דידקטי לא קטן. הפתרון שנמצא לצורך פרויקט זה הוא שימוש באובייקט שנקרא Blob-Binary Large Object, שזהו אובייקט של java כך שקיים ממשק טבעי בשרת apache-derby
* עוד בעיה היתה לאפשר למערכת ל**שלוח הודעות בזמן אמת** לכל המשתמשים המחוברים, מכיוון שהשימוש העיקרי בתעבורה בין לקוח לשרת מתבסס על servlets , שזאת טכנולוגייה הודעות-תגובות, היה צריך למצוא מנגנון ששומר על "קשר" מסויים בין לקוח-שרת, קשר זה נקרא ב Java בשם Session והוא מאפשר שמירה של נתוני הלקוח כל עוד הוא מחובר לשרת. סכימה של session אפשר לראות להלן:

Client 1

Client 2

server

servlet

HttpRequest

HttpRequest

HttpSession

123

HttpSession

134

web

DB

כל לקוח מקבל מספר מזהה ל session שלו ופרטי הלקוח נשמרים כל עוד מתקיים אחד מהשניים:

1. הלקוח לא סיים את ההתקשרות, קרי, לא נשלחה לשרת הודעת session.close() מהלקוח.
2. השרת סיים ניתק את הקשר (connection) עם הלקוח.

אבל, למעשה מה שהוצג לעיל אינו אובייקט Session אלא אובייקט מעט שונה שנקרא HttpSession וזהו אובייקט המשמש לשמירת נתוני ה session כאשר ההתקשרות היא דרך servlet. ה Session הוא אובייקט של הפלטפורמה JEE ,לכן, בשונה מהאובייקט HttpSession מאפשר שימוש גם מעל פרוטוקול התקשרות שנקרא websocket, כמוראה להלן:

Client 1

Client 2

server

websocket

HttpRequest

HttpRequest

Session

123

Session134

web

DB

ההבדל הגדול בין שתי צורות התקשרות הוא שבתקשורת websocket אין מנגנון דגימה (polling) כלומר, השרת לא בודק שהקשר עם הלקוח "חי" או לא: אם השרת ישלח הודעת ping הוא יקבל תשובת pong. כך, תחת פרוטוקול HTTP אפשר להעביר הודעות בין השרת ללקוח ולהיפך בצורה חלקה. בפרוטוקול websocket קיימת "לחיצת יד" בין לקוח ושרת ולאחר מכן אפשר להעביר הודעות בין הצדדים *שלא בצורת הודעה-תגובה*.