גרסה 2 - רכיב ייצוג, התראות זמן אמת, והגדרות מדיניות - 2020

כל גרסאות הפרויקט מתייחסות אל ותלויות במסמך הדרישות הכללי ובמסמך מתודולוגיית פיתוח הפרויקט. כל גרסה מרחיבה או משנה את הגרסה שקדמה לה.

גרסה 2 עוסקת בהעשרת המערכת ברכיב ייצוג (presentation) התומך בהתראות זמן אמת. דרישות הקיבול והזמינות מהמערכת מכתיבות בחירה בארכיטקטורת שרת-לקוח, כאשר רכיב שכבת השירות שנבנה בגרסה 1 משמש כמנשק בין רכיב הייצוג לבין רכיב ה-domain. בגרסה זו, מוגדרים סוגי קנייה והנחה ומדיניות קנייה והנחה, ומועשרים באופני תלויות חדשים.

תפקידי צוות: מנהל/ת גרסה המשמש גם כבודק מטעם הלקוח, ומפתחים. פירוט תחומי האחריות של התפקידים השונים נמצא במסמך המתודולוגיה. במיוחד יש לשים לב לתחומי האחריות של מנהל הגרסה.

דרישות נוספות לגרסה

1. **דרישות רמת שירות:**

יש לממש רכיב ייצוג (presentation layer) שמותאם להצגה בדפדפנים. כתוצאה מכך, יש לקיים מספר דרישות רמת שירות רלוונטיות:

a. **דרישה 2 - אבטחה:** בשלב זה, עיקר התמיכה בדרישה יבוא לידי ביטוי בשימוש בפרוטוקולי תקשורת מאובטחים (HTTPS/WSS) בתקשורת בין רכיב הייצוג לבין המערכת.

b. **דרישה 3c - ממשק משתמש נוח:** הנחיות ועצות למימוש דרישה זו מופיעות בהמשך, בחלק העוסק בפירוט במידול ובמימוש.

c. **דרישה 3d - ממשק המשתמש תואם את תפקיד המשתמש במערכת:** הממשק יחשוף לכל משתמש רק את הפעולות אותן הוא רשאי לבצע בכל הקשר במערכת, בהתאם לתפקידי המשתמש ולהרשאותיו.

d. **דרישה חדשה 3e - המערכת מספקת הסברים למשתמש לגבי פעולותיה:** למשל, אם פעולת קנייה נכשלת, המערכת נדרשת להציג למשתמש מידע אודות סיבת הכישלון (הפרה של מדיניות קנייה, מערכת גביית הכספים דחתה את התשלום וכד'). כמו כן, על המערכת להודיע למשתמש גם במקרים בהם הפעולות אותן ביקש לבצע הסתיימו בהצלחה.

2. **דרישות פונקציונאליות:**

a. **דרישה 4.2 -** הגדרה ועריכה של סוגי הנחה ורכישה בחנות לבעלי הרשאות מתאימות.

בנוסף, גרסה זו מרחיבה את הפעולות של המערכת, אך אין תוספת של תמיכה בתרחישי שימוש חדשים, מעבר לגרסאות הקודמות. הפעולות החדשות מופיעות במספר תרחישי שימוש. יש לעדכן את תרחישי השימוש הקיימים, את מימושם וכן את המודלים הקיימים, בהתאם.

b. **התראות זמן אמת –** דרישה 10**: a** התראות זמן אמת אינן תרחיש שימוש, אלא אפיון של התנהגות המערכת בהקשרים שונים, כפי שמפורט במסמך הדרישות הכללי. יש לבדוק את סיפורי השימוש הרלבנטיים, ולוודא שהם מפרטים את הביצוע של התראות זמן אמת.

   i. יש לשים לב לדרישת ההתראות המושהות: ההתראות של מנויים שאינם מחוברים למערכת בזמן ההתראה, נשמרות במערכת ומוצגות להם בעת ביקורם הבא.

   ii. קבוצות של שישה סטודנטים נדרשים לתמוך גם בדרישה **e10** (התראה למנוי כאשר קיבל תגובה להודעות – שאלות או פניות – שלו לחנות).

c. **הנחות בקניה:** בגרסה זו נוספת תמיכה בסוגי הנחות, ובכללים להפעלתן – מדיניות .

   i. **סוגי הנחות:** גלויה ומותנית. יש להבהיר את מהות הפעולה של חישוב הנחה: קלט, פלט, תנאי קדם. של הנחה מותנית.

1. הנחה גלויה מוגדרת עבור מוצר, וללא התניה.
2. הנחה מותנית יכולה להיות מוגדרת ברמת המוצר ("קנה 2 ביסלי וקבל את השלישי מתנה") או ברמת החנות ("5% הנחה בקופה על סל קניות בשווי לפחות 200 שקלים").

ii. **מדיניות הנחה:** כללים להפעלת הנחות יהיו כללי הרכבה או התניה. כלומר, ייצרו **הנחות מורכבות** שמבוססות על הנחות פשוטות יותר. כלומר כללי ההנחה מייצרים פעולות הנחה מורכבות (לא פרדיקטים)!

1. דוגמאות של הנחות מורכבות:

- "אם אתה זכאי להנחה על גבינת קוטג' אז אתה זכאי גם להנחה של 50% על חלב".
- "הנחה על מוצרי חלב או מאפים, אבל לא שתיהן" – אין כפל מבצעים.
- תנאי הנחה מורכב: למשל, "קנה מסמרים ב-60% הנחה בתנאי שקנית מסור-דיסק או (פטיש וגם מברג) ללא הנחת מועדון חברים".

2. ניתן ליצור הנחה מורכבת מהרכבת הנחות מותנות ו/או גלויות באמצעות אופרטורים הלוגיים "וגם", "או" ו-"או בררני" (XOR). דוגמאות להרכבה: "אין כפל מבצעים",

תבנית העיצוב ***composite***, המאפשרת בנייה היררכית של ישויות, יכולה להיות לעזר. הקבוצה יכולה לקבל החלטות פרטניות לגבי סדר הפעלת ההנחות וכד'.

3. הגדרת הנחות מורכבות ותנאיהן ניתנת להגדרה באמצעות הממשק הגרפי. כלומר, הממשק מאפשר שימוש באופני ההרכבה בצורה גמישה, לבעלי הרשאת מתאימות.

d. מדיניות קנייה: אלו כללים (פרדיקטים) הקובעים אלו פעולות קנייה מותרות.

i. מדיניות קנייה יכולה להיות מוגדרת עבור כל החנות ("ניתן לרכוש לכל היותר 5 מוצרים בקנייה יחידה") או עבור מוצר בודד ("ניתן לרכוש לכל היותר חמישה צמיגים ולכל הפחות שני צמיגים").

ii. מדיניות קנייה יכולה להתחשב בפרטי מוצר ("קניית ביצים מוגבלת לקרטון בודד בכל קנייה"), פרטי סל קניות ("ניתן לרכוש מהחנות לכל היותר 5 מוצרים בקנייה יחידה"), פרטי מערכת ("לא ניתן לקנות מהחנות בימי שבת"), או פרטי משתמש ("לא ניתן למכור למשתמשים מחוץ לישראל").

iii. בדומה להנחות מורכבות, יש להגדיר ישויות שהן **כללי קנייה המורכבים** באמצעות האופרטורים "וגם", "או" ו"או בררני".

iv. הגדרת כללי קנייה מורכבים ניתנת להגדרה באמצעות הממשק. כלומר, הממשק מאפשר שימוש באופני ההרכבה בצורה גמישה, לבעלי הרשאת מתאימות.

מידול ומימוש

1. **רכיב הלקוח (Interface layer):** את רכיב הלקוח יש לממש כ- Web client שישמש כממשק גרפי (GUI) המסוגל לקבל ולהציג התראות למשתמש בזמן אמת.

a. **מידול ממשק המשתמש יעשה באמצעות מודל מצבים היררכי (*Statecharts*)**: רכיב הלקוח מאופיין במצבים המגיבים לאירועים (events) מבחוץ. על כן, טוב לתאר את פעולתו באופן מופשט על ידי statechart. **לכל סיפור שימוש יהיה מצב היררכי ייעודי,** ואולי מצבים היררכיים מקוננים נוספים. ההיררכיה חיונית לצורך הפרדה ושליטה. הממשק הגרפי (GUI) ימומש בהתאם ל-statechart באופן ישיר.

b. **ממשק משתמש נוח:**

i. לצורך זה יש לכתוב מסמך המגדיר את המושג ממשק נוח באמצעות אוסף מאפיינים קונקרטיים הניתנים לבדיקה.
דוגמאות:
- מאפיין "המלל צריך להיות קריא" הוא ערטילאי ולא ניתן לבדיקה.
- המאפיי ן "טקסט עצמאי צריך להיות קרי א, על פי המדדים המוגדרים בדרישות רמת שירות (SLI) עבור מטרת ה'קריאות' (SLO)" הוא קונקרטי. המדד יכול למשל להגדיר יישור הטקסט משמאל ומימין, או את הגודל של רווחים בין שורות ביחס לגודל אותיות, כמדדים וכו' .

ii. ממשק המשתמש צריך להתאים לאפיון הכתוב.
iii. יש להגדיר בדיקות (לאו דווקא אוטומטיות) עבור המאפיינים שהוגדרו.

בנספח של גרסה זו מופיעה תזכורת לגבי המאפיינים של דרישת-תוכנה כתובה היטב.

2. **התראות זמן אמת:** רכיב הלקוח הוא חלק מהארכיטקטורה השכבתית, המאופיינת בתקשורת חד כיוונית. התראות זמן אמת סותרות את עקרון חד-הכיוונית השיכבתי, מכיוון שזוהי תקשורת מכוונת "החוצה", משכבת ה-domain אל רכיב הלקוח (דרך רכיב התקשורת). סוג כזה של תקשורת מטופל על ידי תבנית העיצוב **Observer**. במערכת קיים **publisher** המתחזק אוספי לקוחות מסוגים שונים, ואחראי על שליחת התראות לפי סיווגים שונים ללקוחות. ההתראות הן תמציתיות בתכנן, וגורמות ללקוחות לפנות למערכת לקבלת המידע הרלוונטי. שימו לב להפריד בין ה-**publisher** לבין המרכיב הטכנולוגי של התקשורת.

מומלץ להשתמש בפרוטוקול WebSockets עבור המימוש של התראות זמן אמת; הקדישו מחשבה אילו מהבקשות צריכות להישלח באמצעות WebSockets ואילו לא.

3. **ארכיטקטורה:** הארכיטקטורה העדכנית של המערכת תכלול את השכבות הבאות:
- רכיב לקוח (**presentation layer**): זהו ה-Web Client שאותו יש לממש. על מנת לתמוך בתקשורת בין רכיב הלקוח לבין השרת.
- יש להוסיף **רכיב תקשורת**. רכיב התקשורת יהיה אחראי להעביר את הפניות וההודעות המגיעות מרכיב הלקוח ואליו. על רכיב זה לתמוך בשני אופני התקשורת מ- ואל המערכת (HTTPS ו-WSS).
- שכבת שירות (**service layer**) עדכנית.
- שכבת **domain** עדכנית.

תוצרי גרסה 2

מסמך המתודולוגיה מתאר את התוצרים השונים ואופן תיאורם. התוצרים בגרסה זו מבוססים על התוצרים מגרסה 1 ומרחיבים עליהם. יש להקפיד על תיאום מוחלט בין מודלים למימוש, ולהציג באופן בולט את כל השינויים וההוספות שנעשו ביחס לגרסה הקודמת. מנהל הגרסה אחראי על הצגת תוצרי הגרסה.

1. דו"ח גרסה:
a. פירוט ההספק לגרסה זו: אילו משימות מוכנות במלואן, אילו באופן חלקי ואילו נדחו לגרסה עתידית.
b. תיעוד ניהול הגרסה, ע"פ העקרונות המוצגים במסמך המתודולוגיה (2.1.1).

2. **תיקונים** מגרסה 1, במידת הצורך ובהתאם להנחיות המנחה.

3. מודלים עדכניים:
a. מילון מונחים עדכני.
b. תרחישי שימוש עדכניים.
c. ארכיטקטורה עדכנית.
d. מודל מחלקות עדכני (דיאגרמה לבנה).

4. אפיון ומידול **ממשק המשתמש**:
a. מסמך דרישות לממשק המשתמש

b. מודל ממשק המשתמש כמודל מצבים היררכי (statechart)

5. מימוש:

a. שכבות ה-domain והשירות (service) באופן התואם למודל המחלקות, למודל הארכיטקטורה, ולתרחישי השימוש.

b. שכבת הייצוג (presentation) באופן התואם את ה-statechart ההיררכי, ואת מסמך האפיון שהכנתם.

c. מימוש מפורש לתמיכה בהתראות זמן אמת, על פי תבנית העיצוב Observer.

6. בדיקות: הוספה ועדכון בדיקות לפי הדרישות החדשות. עבור כל הסוגים, כולל טסטים לממשק הגרפי ע"פ הבדיקות שהגדרתם במסמך האפיון. לא יתקבל מימוש ללא רכיב בדיקות משמעותי!

מצגת כיתה: עקביות ושמירת נתונים לאורך זמן

הגרסה הבאה של המערכת תאפשר גיבוי נתונים לאורך זמן באמצעות רכיב מסד נתונים.

- בחר/י ספרייה המספקת אבסטרקציה של מודל מסד הנתונים ומאפשרת קישור בין המערכת לבסיס נתונים. למשל, עבור מסד נתונים עם מודל טבלאי, תרצו לבחור ספריית ORM שתתרגם בין המודל מונחה העצמים לבין המודל הטבלאי. כלים כאלה מאפשרים למפתח להגדיר מיפוי של ישויות של המערכת לישויות של בסיס הנתונים, ומטפל בשמירה ואחזור של מידע בבסיס הנתונים.
- הסבר והדגם כיצד הספרייה משמשת כגשר בין המערכת לבין בסיס הנתונים (Data access layer).
- הדגם עבודה עם הכלי.
- הסבר והדגם את אופן עבודת הספרייה עבור מצבים של אי-תאימות בין האפליקציה לבין הספרייה:
  - התמודדות עם היררכיית מחלקות.
  - התמודדות עם ההקשר (context) של אובייקטים: שמירת הקשרים בין אובייקטים מסוגים שונים: persistent ל non-persistent ולהיפך.

נספח: עקרונות לדרישה מוגדרת היטב

דרישה היא הגדרת מאפיין או אילוץ של התוכן ההכרחיים לקבלת המוצר או התהליך. מבחינת מאפיין - הדרישה משפיעה על יכולות התוכנה, כאשר תוכן הדרישה משפיע על אופן מימוש התוכנה. מבחינת אילוץ – הדרישה היא הגבלה כלשהי על המערכת – עקב מצב קיים או דרישה עתידית.

התכונות הנדרשות של דרישה טובה הינן:

- <u>נכונות</u> -כל דרישה צריכה לייצג את הפונקציונאליות הנדרשת עבור המערכת.
- <u>בדידות ומזוהות</u> – אפשרות לקרוא כל דרישה באופן עצמאי, וזיהוי ייחודי וחד-ערכי של כל דרישה.
- <u>חד משמעיות</u> - על דרישה להיות ניתנת לפירוש בדרך אחת בלבד. כל דרישה צריכה להיות קצרה, מפורשת וברורה. יש להשתמש במילון מלווה כאשר אנו משתמשים בביטוי שיכולים להיות לו מספר משמעויות.
- <u>שלמות</u> - על הדרישות לכסות בצורה מוגדרת היטב את כל היבטי התוכנה.
- <u>עקביות</u> - יש למנוע הגדרת דרישות הסותרות זו את זו.
- <u>סתירות לוגיות</u> – יש להימנע ממצב של סתירה לוגית, כגון אירוע מפעיל כתנאי קדם לתהליך מסוים שאמור להתרחש בו זמנית עם אותו תהליך.
- <u>עקבות למקור</u> - יש לזהות את מקורה של כל דרישה: דרישה מפורשת או דרישה נגזרת.
- <u>הימנעות מתוכן</u> – יש לוודא שאילוצי/הנחיות התוכן מהווים צורך אמיתי.
- <u>בדיקתיות/ מדידות</u> – נדרשת קביעה מפורשת כיצד יהיה ניתן להוכיח את העמידה בדרישה, בדרך שתיקח זמן סופי. על כל הדרישות להיות מובנות הן על ידי הלקוח והן על ידי המפתחים.