פרויקט מלווה
מערכת ניהול קופונים

27128-822-09

JOHN BRYCE
תלמד הי-טק. זה עובד!
a matrix company

פרויקט מלווה

מערכת ניהול קופונים

קורס 822

הסבה לתכנות Java לסביבות

Client, Enterprise ו-Big Data

תוכן העניינים

כללי

במסגרת הקורס משולב פרויקט מלווה המפורט במסמך.

הפרויקט מחולק לשלושה שלבים:

שלב 1: בניית ליבת המערכת. בשלב זה יוקם 'המוח' של המערכת. ליבת המערכת תהיה אחראית על קליטת נתונים, אכסונם במסד נתונים וניהולם. בשלב זה תיושמנה מיומנויות התכנות והשימוש בתחביר בסיסי על מנת לייצר מערכת עובדת ולוגיקה משמעותית.

שלב 2: שכתוב מערכת הליבה לטכנולוגיות עדכניות בתעשייה. בשלב זה יתבצע מעבר לפלטפורמות מתקדמות ועדכניות בתעשייה כדוגמת Spring ו-Hibernate ושדרוג של הקוד.

שלב 3: שילוב יכולות Web וממשקי משתמש. שלב זה יתווסף לליבת המערכת ויאפשר ללקוחות להשתמש בה באמצעות דפדפנים. שלב זה כולל חשיפת הליבה ל-Web ובניית אתר אינטרנט בטכנולוגיות העדכניות ביותר בתעשייה על מנת להפוך אותה לזמינה עבור משתמשי קצה.

חלה חובת הגשה על כל שלושת השלבים.

תיאור המערכת

מערכת ניהול קופונים מאפשרת לחברות (Companies) לייצר קופונים כחלק מקמפיינים פרסומיים ושיווקיים שהן מקיימות.

למערכת יש גם לקוחות רשומים (Customers). הלקוחות יכולים לרכוש קופונים. קופונים מוגבלים בכמות ובתוקף. לקוח מוגבל לקופון אחד מכל סוג.

המערכת מתעדת את הקופונים שנרכשו ע"י כל לקוח.

הכנסות המערכת הן מרכישת קופונים ע"י הלקוחות ומיצירה ועדכון קופונים חדשים ע"י החברות.

הגישה למערכת מתחלקת לשלושה סוגי Clients:

1. Administrator – ניהול רשימת החברות ורשימת הלקוחות.
2. Company – ניהול רשימת קופונים המשויכים לחברה.
3. Customer – רכישת קופונים.

הנחיות כלליות

- העבודה על הפרויקט יכולה להתבצע באופן עצמאי או בקבוצות של עד שלושה מתכנתים.
- מועדי הנחיה והגשה – כל שלב בפרויקט מוגש עד לתאריך הנחיית השלב הבא. מועדי הצגת הפרויקט מופיעים בתוכנית הקורס. לוח זמנים זה קשיח ולשינויו נדרש אישור מרכזת הקורס או מהגורמים המקצועיים. בכל מקרה של בקשה לשינוי בלוח הזמנים יש לקחת בחשבון הורדת ציון.
- יש להציג את הפרויקט עובד, ללא שגיאות קומפילציה וללא קריסות.
- יש להגיש את קבצי המקור ע"י שליחת קבצים באינטרנט.
- יש להשתמש בתיעוד. רצוי לייצר Java Docs.
- יש להגיש בכל שלב מסמך טקסט מפורט לגבי התקנה, שמות משתמשים וסיסמאות, נתוני התחברות למסד הנתונים והנחיות למשתמש.
- במידה והתווספו יכולות מעבר לדרישות ו/או נעשה שימוש ב-APIs מעבר לנלמד בכיתה – הדבר מבורך ויש לציין רשימת תוספות אלו עם ההגשה על מנת שיתייחסו לכך בבדיקה.
- אין להגיש פרויקט אשר מדפיס Stack Trace במקרים של Exceptions. יש לייצר Exceptions מותאמים למערכת ולהשתמש בהודעות ברורות וקריאות למשתמש בכל מקרה של שגיאה.

שלב 1

בניית ליבת המערכת

Threads ,JDBC ,DAO ,Java Beans ,OOP

תיאור:

בשלב זה יוגדר מסד הנתונים לאכסון ושליפת מידע אודות לקוחות, חברות וקופונים. מעל מסד הנתונים תוקם שכבת בידוד בשם DAO (Data Access Objects) שתאפשר עבודה נוחה מ-Java אל מסד הנתונים.

כמו כן, יוקמו שירותי תשתית בסיסיים כגון מאגר קישורים למסד הנתונים (ConnectionPool) ו-Job יומי המתחזק את המערכת ומנקה אותה מקופונים שפג תוקפם.

יוגדרו שלושה Entry Points למערכת עבור כל אחד מסוגי ה-Clients של המערכת – אדמיניסטרטור, חברה ולקוח, אשר יתחברו ע"י ביצוע Login.

להלן שלבי הביצוע עבור שלב 1:

חלק א' – הגדרת מסד הנתונים ובניית הטבלאות.

חלק ב' – בניית מחלקות Java Beans המייצגות את המידע שבמסד הנתונים.

חלק ג' – בניית ConnectionPool המאפשר ניהול מאגר ה-Connections למסד הנתונים.

חלק ד' – בניית מחלקות ה-DAO המאפשרות ביצוע פעולות CRUD כלליות על מסד הנתונים.

חלק ה' – בניית הלוגיקה העסקית הדרושה ע"י שלושת סוגי ה-Clients של המערכת.

חלק ו' – בניית מחלקה המאפשרת כניסת Clients ולפיכך שימוש בלוגיקה העסקית המתאימה.

חלק ז' – בניית Job יומי למחיקת קופונים שפג תוקפם מהמערכת.

חלק ח' – בניית מחלקת Test להדגמת יכולות המערכת והפעלתה מה-main.

דגשים:

- חשוב לתעד את הקוד.
- חשוב להשתמש בשמות משמעותיים עבור משתנים/פונקציות/מחלקות/Packages וכדומה.
- חובה לכתוב באופן יעיל, ללא העתקת קוד (Don't Repeat Yourself – DRY).
- יש להשתמש ב-Custom Exceptions (לדוגמה: CouponAlreadyExistsException) עבור חריגות ספציפיות של המערכת.
- יש להגדיר try-catch כללי במחלקת ה-Test המסוגל לתפוס את כל ה-Exceptions.
- מומלץ לעבוד עם מסד הנתונים MySQL.
- חובה לחלק את המחלקות ל-Packages מתאימים, בעלי שמות משמעותיים.

חלק א' – הגדרת מסד הנתונים ובניית הטבלאות

להלן רשימת הטבלאות הדרושות:

1. COMPANIES – טבלת חברות:

- ID (int – מפתח ראשי, מספור אוטומטי)
- NAME (string – שם החברה)
- EMAIL (string – אימייל החברה)
- PASSWORD (string – סיסמת כניסה)

2. CUSTOMERS – טבלת לקוחות:

- ID (int – מפתח ראשי, מספור אוטומטי)
- FIRST_NAME (string – שם פרטי)
- LAST_NAME (string – שם משפחה)
- EMAIL (string – אימייל הלקוח)
- PASSWORD (string – סיסמת כניסה)

3. CATEGORIES – טבלת קטגוריות של קופונים:

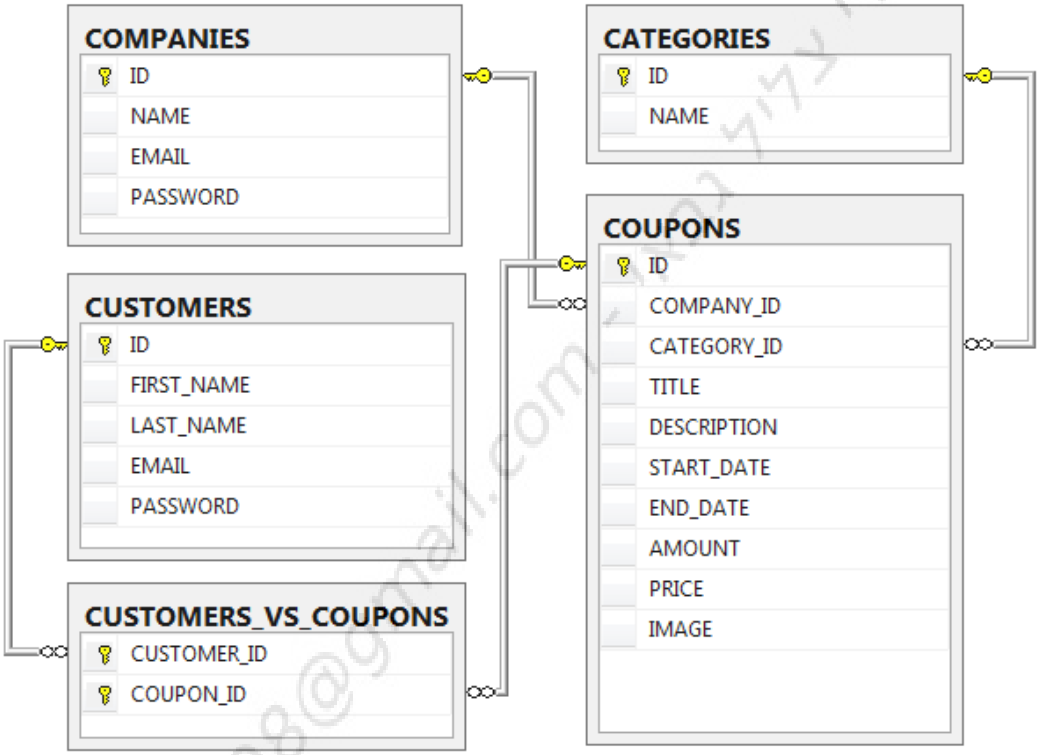
- ID (int – מפתח ראשי, מספור אוטומטי)
- NAME (string – שם קטגוריית הקופון – מזון, מוצרי חשמל, מסעדות, נופש וכו')

4. COUPONS – טבלת קופונים:

- ID (int – מפתח ראשי, מספור אוטומטי)
- COMPANY_ID (int – מפתח זר לקוד החברה בטבלת החברות)
- CATEGORY_ID (int – מפתח זר לקוד הקטגוריה בטבלת הקטגוריות)
- TITLE (string – כותרת הקופון)
- DESCRIPTION (string – תיאור מפורט של הקופון)
- START_DATE (date – תאריך יצירת הקופון)
- END_DATE (date – תאריך תפוגת הקופון)
- AMOUNT (integer – כמות קופונים במלאי)
- PRICE (double – מחיר הקופון ללקוח)
- IMAGE (string – שם קובץ תמונה עבור הקופון)

5. CUSTOMERS_VS_COUPONS – טבלת לקוחות מול קופונים – מאפשרת לדעת אלו קופונים רכש כל לקוח ואלו לקוחות רכשו כל קופון:
   - CUSTOMER_ID (int – מפתח ראשי בטבלה זו ומפתח זר לקוד הלקוח בטבלת הלקוחות)
   - COUPON_ID (int – מפתח ראשי בטבלה זו ומפתח זר לקוד הקופון בטבלת הקופונים)

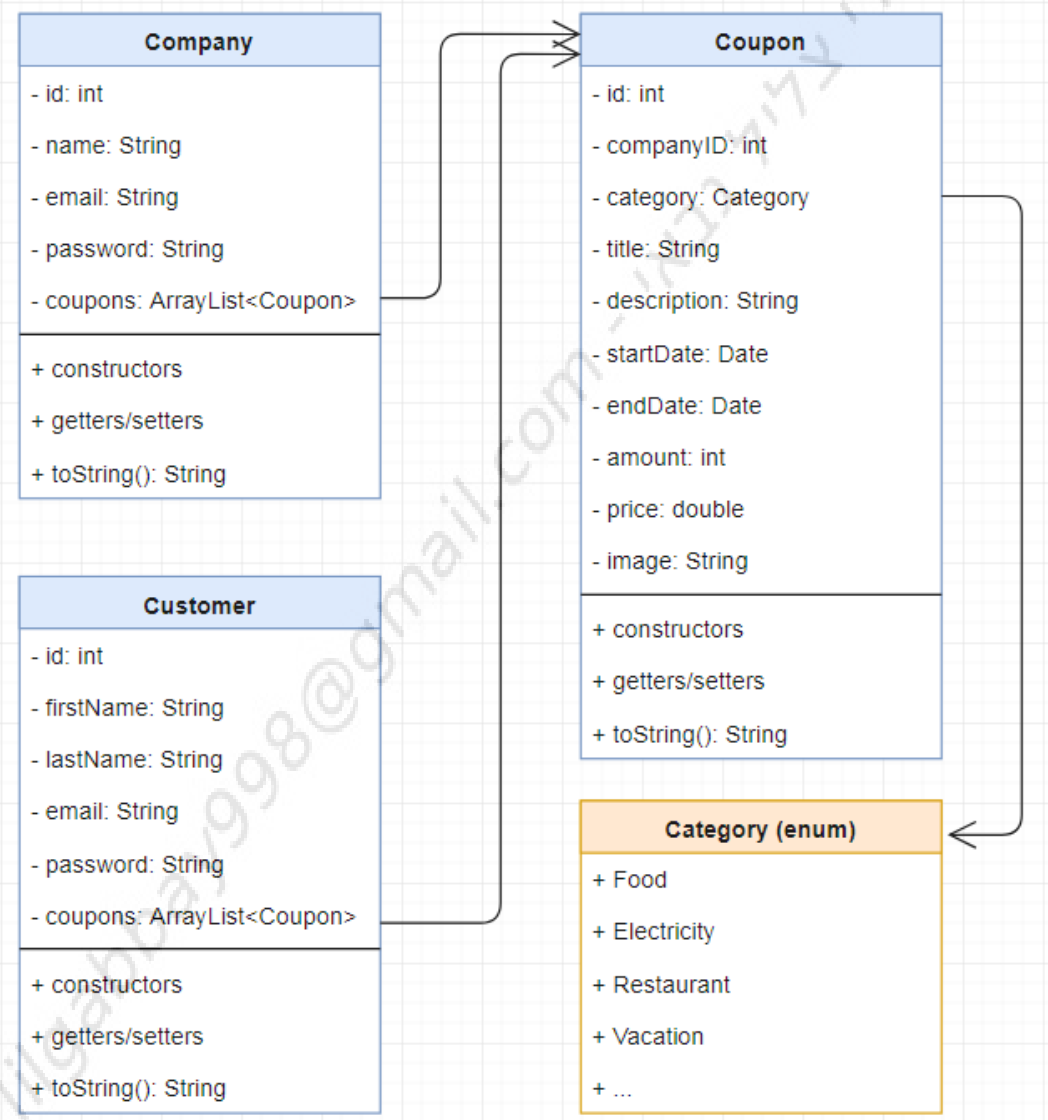
להלן סכמת הטבלאות והקשרים ביניהן:



חלק ב' – בניית מחלקות Java Beans המייצגות את המידע שבמסד הנתונים

Java Beans הינן מחלקות מידע טהור המייצגות את המידע המנוהל ע"י האפליקציה. מחלקות ה-DAO לדוגמה (Data Access Objects), מקבלות אובייקטי Java Beans ומתרגמות אותם לשאילתות SQL שנשלחות למסד הנתונים. כל טבלה עיקרית מיוצגת ע"י מחלקה משלה. הקשרים בין הטבלאות מיוצגים ע"י Collections מתאימים.

להלן סכמת מחלקות ה-Java Beans:



חלק ג' – בניית ConnectionPool המאפשר ניהול מאגר ה-Connections למסד הנתונים

ConnectionPool הינה מחלקת Singleton (מחלקה ממנה קיים אובייקט אחד ויחיד) המאפשרת לנהל מספר קבוע של Connections למסד הנתונים. ה-Connections שמורים במאגר (אוסף מסוג Set) ברמת המחלקה.

להלן המתודות הדרושות במחלקה:

Connection getConnection()

מוציאה מהמאגר אובייקט Connection אחד ומחזירה אותו ע"י return כך שניתן יהיה להשתמש בו לצורך ביצוע פעולות על מסד הנתונים. במידה והמאגר ריק כי כל ה-Connections כרגע בשימוש – מבצעת wait בכדי להמתין עד ש-Connection כלשהו יתפנה ויוחזר למאגר.

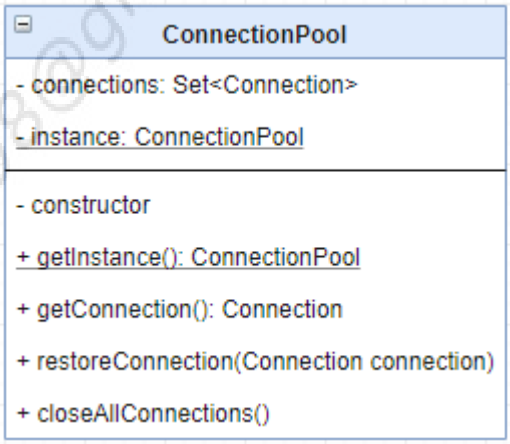
void restoreConnection(Connection connection)

מקבלת כארגומנט אובייקט Connection אחד שהתפנה ומחזירה אותו למאגר. בגלל שכעת התווסף למאגר Connection נוסף – מבצעת notify בכדי להודיע ל-Thread כלשהו שממתין (כלומר ביצע wait) שכעת Connection אחד התפנה וכך ניתן לנסות לקבל אותו.

void closeAllConnections()

סוגרת את כל ה-Connections מבחינת מסד הנתונים.

להלן סכמת מחלקת ה-ConnectionPool:

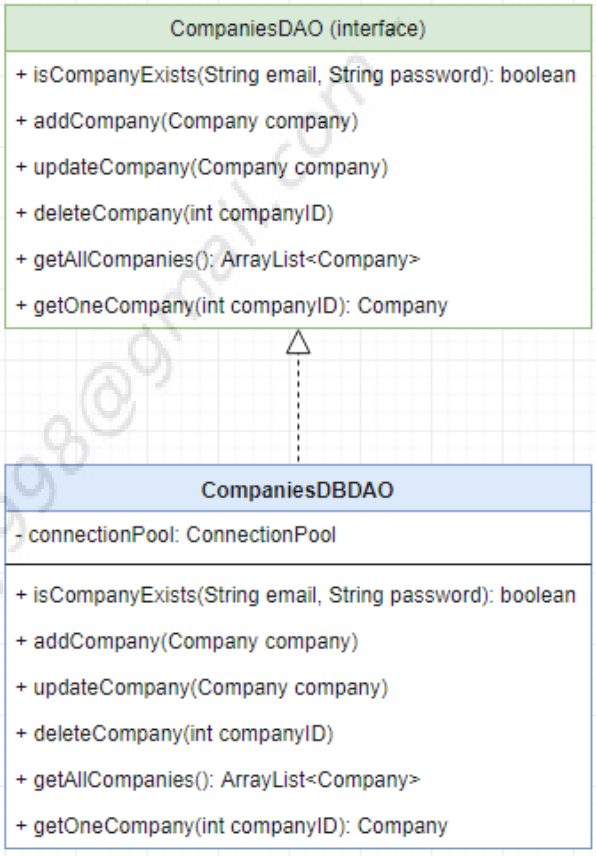


חלק ד' – בניית מחלקות ה-DAO המאפשרות ביצוע פעולות CRUD כלליות על מסד הנתונים

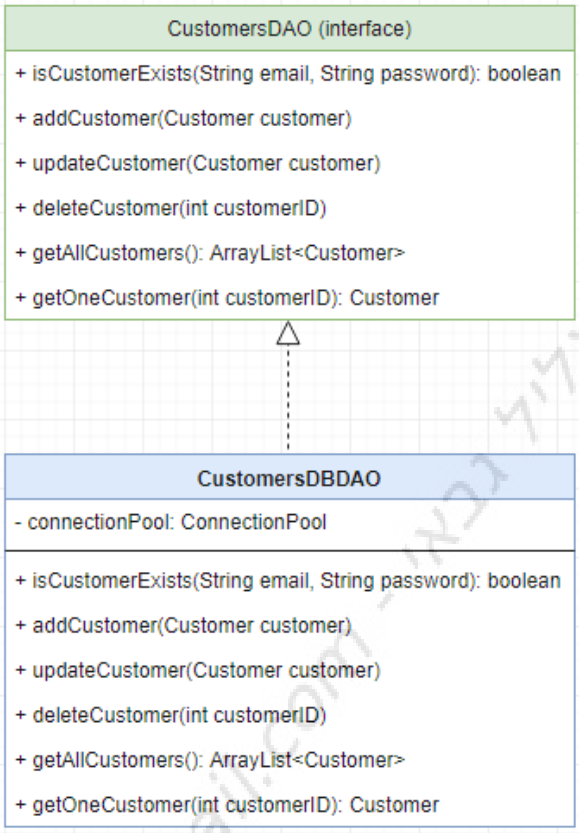
מחלקות DAO (Data Access Objects) אלו מחלקות המאפשרות לבצע פעולות CRUD (Create/Read/Update/Delete) כלליות על הטבלאות במסד הנתונים. מחלקות אלו לא מבצעות את הלוגיקה הקשורה לאפליקציה אלא אך ורק פעולות CRUD כלליות. מחלקות אלו מקבלות כארגומנטים אובייקטי Java Beans או ערכים פרימיטיביים פשוטים (int, string וכדומה), מייצרות מהן שאילתות SQL ומוציאות את השאילתות לפועל במסד הנתונים. כמו כן הן יכולות להחזיר בחזרה אובייקטי Java Beans בודדים, Collections של Java Beans או ערכים פרימיטיביים פשוטים. מחלקות אלו משתמשות ב-ConnectionPool לצורך השגת Connection למסד הנתונים עבור ביצוע הפעולות השונות.

עקב ההפרדה של שכבה זו, אם דרוש יהיה בעתיד מסד נתונים אחר מהקיים, ניתן יהיה להחליף את מסד הנתונים ללא שום נגיעה בשאר חלקי המערכת אלא אך ורק ע"י החלפת שכבת ה-DAO. לצורך הפרדה מושלמת בין שכבה זו לבין שאר חלקי המערכת, נגדיר interface-ים המכילים את הפונקציונליות הדרושה עבור שכבת ה-DAO ונממש את הפונקציונליות הזו במחלקות ה-DAO השונות.

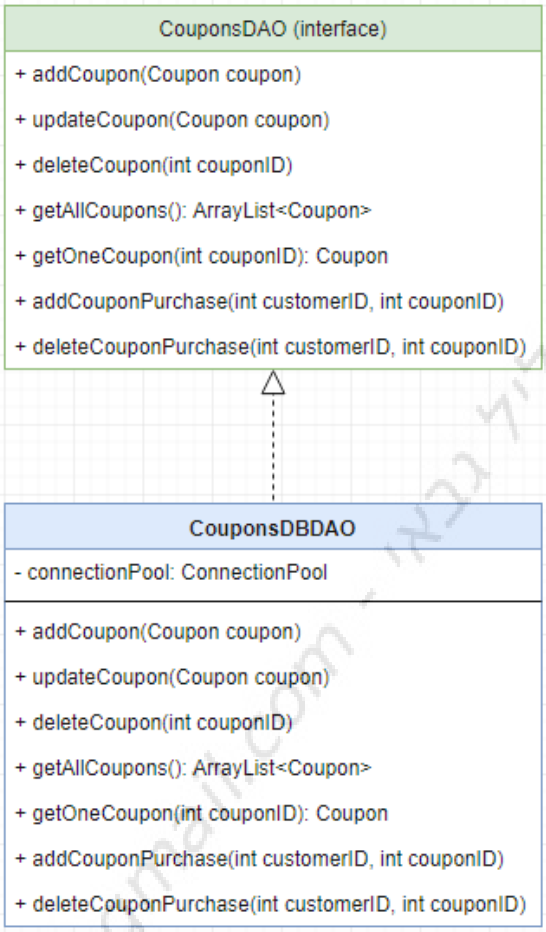
להלן סכמת ה-DAO של טבלת החברות:



להלן סכמת ה-DAO של טבלת הלקוחות:



להלן סכמת ה-DAO של טבלת הקופונים:



- ניתן להוסיף פונקציות CRUD כלליות נוספות ל-interface-ים ולמחלקות הנ"ל אם לדעתכם זה דרוש.
- במקרה של מידע שגוי הנשלח למחלקות אלו, חשוב לזרוק חריגות מתאימות (Custom Exceptions) המתארות את השגיאות בצורה ברורה.

חלק ה' – בניית הלוגיקה העסקית הדרושה ע"י שלושת סוגי ה-Clients של המערכת.

Client נחשב כל מי שיכול להשתמש במערכת. שלושת סוגי ה-Clients של המערכת הינם:

1. Administrator – המנהל הראשי של כל המערכת.
2. Company – כל אחת מהחברות שבמערכת.
3. Customer – כל אחד מהלקוחות שבמערכת.

לכל Client כזה ישנן פעולות לוגיות עסקיות הדרושות לביצוע על ידו. לדוגמה, Administrator יכול להוסיף חברה חדשה, אולם Company או Customer לא יכולים (ולא אמורים) להוסיף חברה חדשה. לדוגמה, Customer יכול לרכוש לעצמו קופון, אולם Administrator או Company לא יכולים (ולא אמורים) לרכוש לעצמם קופון. לכן, כל הפעולות הדרושות לביצוע ע"י כל Client יהיו במחלקה ייעודית המכילה את כל הלוגיקה העסקית הדרושה עבור אותו ה-Client.

כל פעילות לוגית עסקית כזו תתבצע ע"י פונקציה ייעודית ותשתמש במחלקות ה-DAO בכדי לבצע את הפעילות הלוגית שלה. פונקציה כזו, יכולה לצורך ביצוע הפעילות שלה לבצע סדרת פעולות בעזרת מחלקות ה-DAO.

לדוגמה, לצורך רכישת קופון ע"י לקוח (פעולה לוגית אחת שצריכה להיות במחלקת הלוגיקה העסקית של ה-Customers) עלינו לבצע את סדרת הפעולות הבאות ע"י שימוש במחלקות ה-DAO:

א. לוודא שהלקוח לא רכש כבר בעבר קופון כזה.
ב. לוודא שהקופון הדרוש עדיין קיים במלאי (הכמות שלו גדולה מ-0).
ג. לוודא שתאריך התפוגה של הקופון עדיין לא הגיע.
ד. לבצע את רכישת הקופון ע"י הלקוח.
ה. להוריד את הכמות במלאי של הקופון ב-1.

Design כזה, המאפשר להחצין פעולות לוגיות פשוטות, שמאחורי הקלעים משתמשות בסדרת פעולות בסיסיות יותר, נקרא Facade (חזית) שכן לאחר מכן אנו נצטרך רק להשתמש במחלקות ה-Facade הללו לצורך ביצוע הלוגיקה העסקית הדרושה במערכת, ללא שום שימוש נוסף במחלקות ה-DAO הבסיסיות יותר.

שלושת מחלקות הלוגיקה העסקית הינן:

1. AdminFacade – מכילה את הלוגיקה העסקית של Administrator.
2. CompanyFacade – מכילה את הלוגיקה העסקית של Company.
3. CustomerFacade – מכילה את הלוגיקה העסקית של Customer.

בגלל ששלושת המחלקות הללו צריכות להשתמש ברכיבי DAO, יהיה נכון להגדיר מחלקת בסיס (אבסטרקטית כמובן) שמכילה את רכיבי ה-DAO, או לפחות רק References שלהם, ומחלקות ה-Facade השונות יירשו את מחלקת הבסיס הזו וייצרו את רכיבי ה-DAO המתאימים עבורן.

להלן הלוגיקה הדרושה לביצוע ע"י Client מסוג Administrator, שיש לבנות במחלקת ה-AdminFacade:

- כניסה למערכת.
  - במקרה זה (רק עבור Administrator) אין צורך לבדוק את האימייל והסיסמה מול מסד הנתונים, אלא יש לבדוק אותם כ-Hard-Coded.
    האימייל תמיד יהיה admin@admin.com והסיסמה תמיד תהיה admin.
- הוספת חברה חדשה.
  - לא ניתן להוסיף חברה בעלת שם זהה לחברה קיימת.
  - לא ניתן להוסיף חברה בעלת אימייל זהה לחברה קיימת.
- עדכון חברה קיימת.
  - לא ניתן לעדכן את קוד החברה.
  - לא ניתן לעדכן את שם החברה.
- מחיקת חברה קיימת.
  - יש למחוק בנוסף גם את הקופונים שיצרה החברה.
  - יש למחוק בנוסף גם את היסטוריית רכישת הקופונים של החברה ע"י לקוחות.
- החזרת כל החברות.
- החזרת חברה ספציפית לפי קוד חברה.
- הוספת לקוח חדש.
  - לא ניתן להוסיף לקוח בעל אימייל זהה ללקוח קיים.
- עדכון לקוח קיים.
  - לא ניתן לעדכן את קוד הלקוח.
- מחיקת לקוח קיים.
  - יש למחוק בנוסף גם את היסטוריית רכישת הקופונים של הלקוח.
- החזרת כל הלקוחות.
- החזרת לקוח ספציפי לפי קוד לקוח.

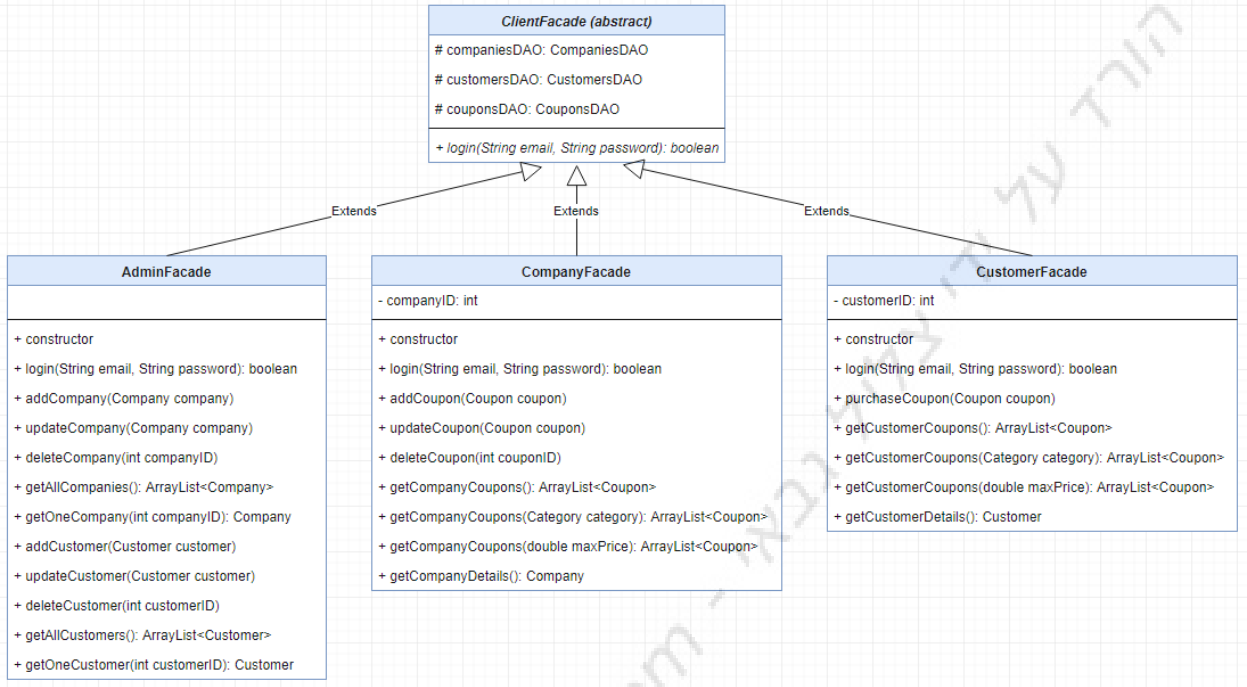
להלן הלוגיקה הדרושה לביצוע ע"י Client מסוג Company, שיש לבנות במחלקת ה-CompanyFacade:

- כניסה למערכת.
  - יש לבדוק את פרטי ה-Login (אימייל וסיסמה) מול מסד הנתונים.
- הוספת קופון חדש.
  - אין להוסיף קופון בעל כותרת זהה לקופון קיים של אותה החברה. מותר להוסיף קופון בעל כותרת זהה לקופון של חברה אחרת.
- עדכון קופון קיים.
  - לא ניתן לעדכן את קוד הקופון.
  - לא ניתן לעדכן את קוד החברה.
- מחיקת קופון קיים.
  - יש למחוק בנוסף גם את היסטוריית רכישת הקופון ע"י לקוחות.
- החזרת כל הקופונים של החברה.
  - כלומר יש להחזיר את כל הקופונים של החברה שביצעה Login.
- החזרת כל הקופונים מקטגוריה ספציפית של החברה.
  - כלומר יש להחזיר רק קופונים מקטגוריה ספציפית של החברה שביצעה Login.
- החזרת כל הקופונים עד מחיר מקסימלי של החברה.
  - כלומר יש להחזיר רק קופונים עד מחיר מקסימלי של החברה שביצעה Login.
- החזרת פרטי החברה.
  - כלומר יש להחזיר את פרטי החברה שביצעה Login.

להלן הלוגיקה הדרושה לביצוע ע"י Client מסוג Customer, שיש לבנות במחלקת ה-CustomerFacade:

- כניסה למערכת.
  - יש לבדוק את פרטי ה-Login (אימייל וסיסמה) מול מסד הנתונים.
- רכישת קופון.
  - לא ניתן לרכוש את אותו הקופון יותר מפעם אחת.
  - לא ניתן לרכוש את הקופון אם הכמות שלו היא 0.
  - לא ניתן לרכוש את הקופון אם תאריך התפוגה שלו כבר הגיע.
  - לאחר הרכישה יש להוריד את הכמות במלאי של הקופון ב-1.
- החזרת כל הקופונים שהלקוח רכש.
  - כלומר יש להחזיר את כל הקופונים שרכש הלקוח שביצע Login.
- החזרת כל הקופונים מקטגוריה ספציפית שהלקוח רכש.
  - כלומר יש להחזיר רק קופונים מקטגוריה ספציפית של הלקוח שביצע Login.
- החזרת כל הקופונים עד מחיר מקסימלי שהלקוח רכש.
  - כלומר יש להחזיר רק קופונים עד מחיר מקסימלי של הלקוח שביצע Login.
- החזרת פרטי הלקוח.
  - כלומר יש להחזיר את פרטי הלקוח שביצע Login.

להלן סכמת מחלקות ה-Facade:

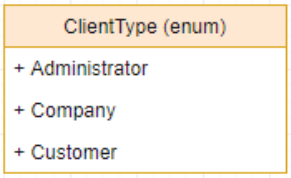


- המשתנה companyID במחלקת ה-CompanyFacade אמור להכיל את קוד החברה שביצעה Login.
- המשתנה customerID במחלקת ה-CustomerFacade אמור להכיל את קוד הלקוח שביצע Login.
- ניתן להוסיף פונקציות עזר נוספות למחלקות הנ"ל אם לדעתכם זה דרוש.
- במקרה של מידע שגוי הנשלח למחלקות אלו, חשוב לזרוק חריגות מתאימות (Custom Exceptions) המתארות את השגיאות בצורה ברורה.

חלק ו' – בניית מחלקה המאפשרת כניסת Clients ולפיכך החזרת מחלקת ה-Facade המתאימה

מחלקה זו הינה מחלקת Singleton בשם LoginManager המכילה פונקציית Login שמאפשרת לכל אחד משלושת סוגי ה-Clients להתחבר למערכת.

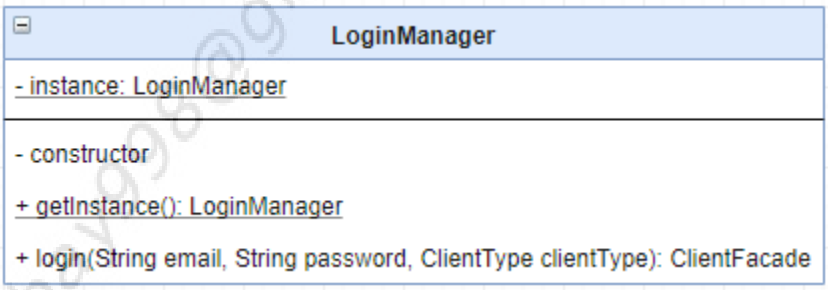
ראשית, יש לבנות Enum בשם ClientType המתאר את כל אחד מסוגי ה-Clients:



שנית, יש לבנות במחלקה LoginManager פונקציית Login שתקבל אימייל, סיסמה ומשתנה מסוג ClientType. על הפונקציה לבדוק האם פרטי ה-Login נכונים בהתאם ל-ClientType. אם פרטי ה-Login שגויים – הפונקציה תחזיר null. אם פרטי ה-Login נכונים – הפונקציה תחזיר את מחלקת ה-Facade המתאימה:

- עבור Administrator שביצע כניסה נכונה – יוחזר אובייקט AdminFacade.
- עבור Company שביצע כניסה נכונה – יוחזר אובייקט CompanyFacade.
- עבור Customer שביצע כניסה נכונה – יוחזר אובייקט CustomerFacade.
- עבור כל כניסה שגויה – יוחזר null.

להלן סכמת מחלקת ה-LoginManager:



חלק ז' – בניית Job יומי למחיקת קופונים שפג תוקפם מהמערכת

Job הינו תהליך שרץ ברקע באופן קבוע ומבצע פעולה כלשהי. Job יכול לבצע את הפעולה שלו ללא הפסקה, או שהוא יכול לבצע אותה בזמנים ספציפיים, לדוגמה פעם בשעה או פעם ביום או פעם בשבוע וכו' – תלוי בפעילות שהוא אמור לבצע. לצורך ביצוע פעולה בזמנים ספציפיים אפשרי לבדוק בכל שנייה/דקה/שעה וכדומה – האם הזמן הדרוש כבר הגיע, ואם כן – לבצע את הפעולה הדרושה.

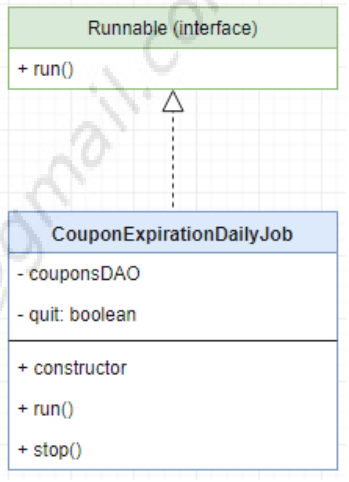
מימוש ה-Job חייב להיות ע"י Thread נפרד, שכן הוא צריך לרוץ ללא הפסקה במקביל לשאר פעילויות המערכת.

בחלק זה דרוש בניית Job יומי, כלומר שיתבצע פעם אחת בכל יום, ושימחק קופונים שפג תוקפם. לצורך מחיקת קופון שפג תוקפו יש למחוק את הקופון מטבלת הקופונים ויש למחוק בנוסף גם את היסטוריית הקנייה של הקופון.

במחלקת ה-Job יש לבנות פונקציה המתחילה את ה-Job ופונקציה הגורמת ל-Job להסתיים.

על ה-Job להתחיל לעבוד בתחילת התוכנית ולהסתיים בסוף התוכנית.

להלן סכמת ה-Job למחיקת קופונים שפג תוקפם מהמערכת:



- יש להפעיל את ה-Job עם עליית המערכת (תחילת התוכנית).
- יש להפסיק את ה-Job עם סגירת המערכת (סיום התוכנית).

חלק ח' – בניית מחלקת Test להדגמת יכולות המערכת והפעלתה מה-main

מחלקה זו נקראת Test ומכילה פונקציה סטטית אחת בשם testAll שתפקידה לבדוק את כל המערכת ע"י קריאה לכל אחת מפונקציות הלוגיקה העסקית שבניתם. אין צורך לקבל דבר מהמשתמש. כל הבדיקות צריכות להיות Hard-Coded. בהמשך – המידע יועבר למערכת ע"י אתר האינטרנט.

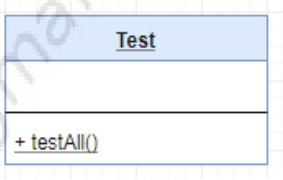
על הפונקציה testAll לבצע את הפעולות הבאות:

א. הפעלת ה-Job היומי.
ב. התחברות ע"י ה-LoginManager כ-Administrator, קבלת אובייקט AdminFacade וקריאה לכל פונקציית לוגיקה עסקית שלו.
ג. התחברות ע"י ה-LoginManager כ-Company, קבלת אובייקט CompanyFacade וקריאה לכל פונקציית לוגיקה עסקית שלו.
ד. התחברות ע"י ה-LoginManager כ-Customer, קבלת אובייקט CustomerFacade וקריאה לכל פונקציית לוגיקה עסקית שלו.
ה. הפסקת ה-Job היומי.
ו. סגירת כל ה-Connections (קריאה לפונקציה closeAllConnections של ה-ConnectionPool).

את כל הפעולות הללו יש להגדיר בתוך try-catch אחד ולהציג את הודעת השגיאה במקרה של חריגה.

אין להציג הודעות של Stack Trace על המסך, אלא רק הודעות שגיאה ברורות המגיעות מה-Custom Exceptions.

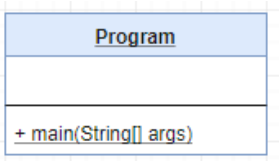
להלן סכמת מחלקת ה-Test:



המחלקה הראשית של המערכת הינה מחלקת ה-Program, המכילה את פונקציית ה-main.

על פונקציית ה-main לבצע קריאה אחת לפונקציה testAll של מחלקת ה-Test.

להלן סכמת מחלקת ה-Program:



שלב 2

שכתוב המערכת תוך שימוש בטכנולוגיית Spring

Hibernate ,JPA ,Spring Boot ,Spring Framework

תיאור:

בשלב זה המערכת למעשה תבנה מחדש כשהיא מתבססת על טהרת Spring ו-Hibernate. ניתן כמובן להעביר קוד משלב 1 של הפרויקט ו"לאמץ" אותו לסביבת Spring. כמובן שבהזדמנות זו מומלץ גם לשכתב מימושים טעוני שיפור.

הפרויקט חייב להתבסס על Spring Boot עם ה-Extensions הבאים:

- Spring JPA
- MySQL Driver או כל Driver אחר המתאים למסד הנתונים שבחרתם.

להלן שלבי הביצוע עבור שלב 2:

1. יש ליישם את רכיבי ה-DAO באמצעות Entity Beans המקבילות ל-Java Beans מהשלב הראשון ובאמצעות Spring Repositories.
2. יש להוסיף Custom Queries ל-Repositories שנבנו באמצעות Spring בהתאם לדרישות השלב הראשון.
3. יש להפוך את מחלקות ה-Facades מהשלב ראשון ל-@Services.
4. יש ליישם את מחלקת ה-CouponExpirationDailyJob כ-Singleton.
5. יש ליישם את מחלקת ה-LoginManager כ-Singleton.
6. יש לבצע בדיקה כללית לכל המערכת ע"י יצירת מחלקה בשם Test המוגדרת גם היא כ-Singleton ובה מתודה המבצעת "תצוגת תכלית" (הצגה ובדיקה של כל המערכת). יש לבצע @Autowired של רכיבי המערכת הרלוונטיים, לדוגמה LoginManager. בנוסף, יש להפעיל את המתודה המבצעת את "תצוגת התכלית" מ-()SpringBootApplication.main כך שהרצת הפרויקט תפעיל בצורה אוטומטית את הקוד במחלקת ה-Test.

שלב 3

בניית RESTful Service באמצעות Spring MVC
ובניית אתר אינטרנט באמצעות Angular

FullStack Development ,Angular ,SPA ,Sprint MVC ,REST

תיאור:

בשלב זה אנו מחצינים את הלוגיקה העסקית של השרת אל העולם כך שניתן יהיה להשתמש בה על גבי רשת האינטרנט, לדוגמה ע"י אתר אינטרנט. הטכנולוגיות העדכניות לכך כיום הינם מימוש RESTful Service בצד השרת ושימוש בו ע"י אתר אינטרנט בטכנולוגיית SPA (Single Page Application).

להלן שלבי הביצוע עבור שלב 3:

חלק א' – יחשפו כל אחד מסוגי ה-Clients (Administrator/Company/Customer) כ-Service בצד השרת. כלומר, יהיו שלושה REST Services אשר יאפשרו אינטגרציה בין צד הלקוח ומערכת השרת.

חלק ב' – יבנה אתר אינטרנט בטכנולוגיית Angular המאפשר ביצוע Login כאדמיניסטרטור, חברה או לקוח וביצוע כלל הפעולות הנתמכות במערכת.

דגשים:

- יש להשתמש ב-Spring MVC עם Embedded Tomcat. זאת ע"י תוספת ב-Starter-Web שבוצע בשלב 2.
- מעבר הנתונים הגולמיים בין הלקוח לשרת יהיה באמצעות JSON בלבד. לא Plain-Text ולא XML.
- חובה לכתוב באופן יעיל, ללא העתקת קוד.
- Exceptions שהוגדרו עבור המערכת יועברו כ-Message ללקוח על מנת שיוצגו במידת בצורך.
- יש להגדיר @RestController עבור כל Facade Service.
- חובה לנהל גישות למערכת באמצעות Tokens אשר יונפקו לכל לקוח עם ביצוע Login. כל נסיון גישה למערכת ללא Token אמור להיחסם ולהפנות את הלקוח לביצוע Login. מומלץ לעשות זאת במחלקה נפרדת בשם TokensManager אשר תנהל רשימה תקפה של Tokens ותהיה בשימוש ה-Controllers.
- יש לאפשר למשתמש לבצע Logout מהמערכת.
- הממשק צריך להיות ברור ונוח לשימוש ויישום בארכיטקטורת SPA ע"י Angular.

חלק א' – הגדרת RESTful Service עבור כל אחד מה-Clients הנתמכים במערכת

בחלק זה יש לבנות מחלקות Controllers עבור כל אחד מה-Clients הנתמכים במערכת:

- AdminController
- CompanyController
- CustomerController

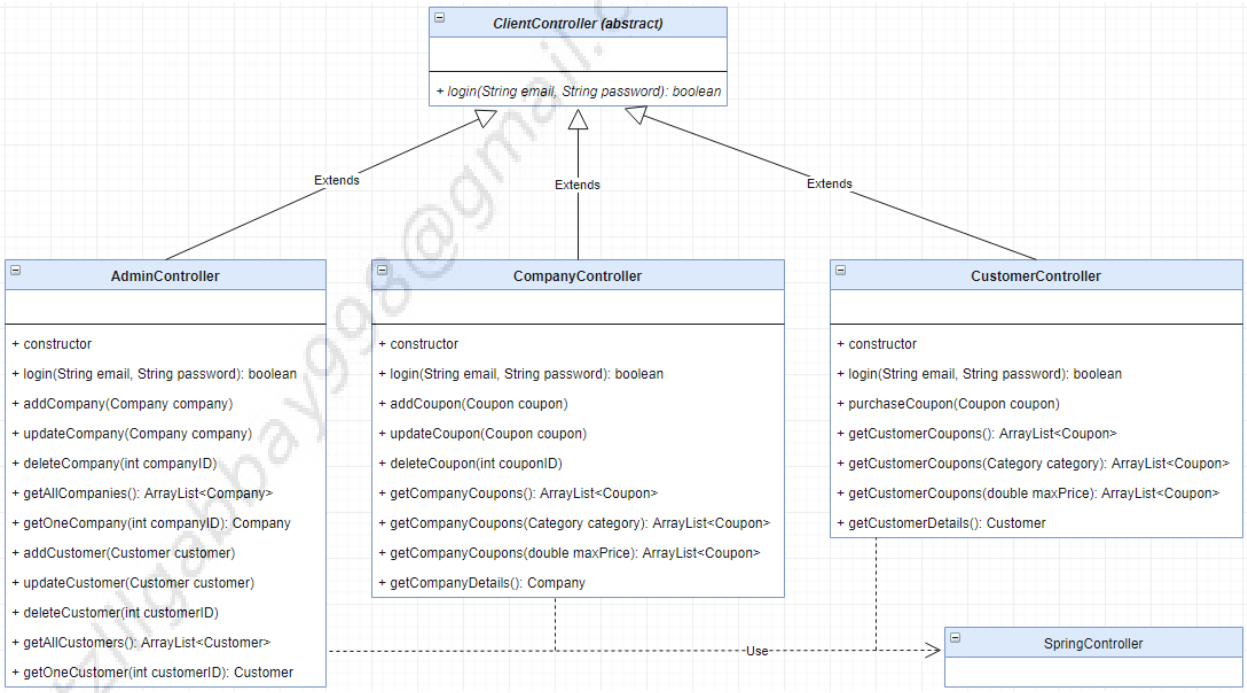
כל מחלקת Controller מאפשרת שליחת וקבלת נתונים ל-Facade הרלוונטי שלה באמצעות חשיפת הפונקציונליות כ-RESTful Service.

עם ביצוע Login ע"י לקוח הקצה, ייווצר Token המקושר ל-Facade הרלוונטי ללקוח.

יש להשתמש במחלקה נפרדת, לדוגמה TokensManager על מנת לנהל את כלל ה-Tokens לכל סוגי ה-Clients. מחלקה זו צריכה לדעת למחוק Token באחד משני מקרים:

- כאשר לקוח הקצה מבצע יציאה מסודרת מהמערכת (Logout)
- כאשר לא התבצעה שום פעילות בצד השרת ב-30 הדקות האחרונות

להלן סכמת רכיבי ה-RESTful Service:



חלק ב' – בניית אתר אינטרנט בטכנולוגיית Angular

בחלק זה יש לבנות אתר אינטרנט בטכנולוגיית Angular הבנוי כ-SPA (Single Page Application) שמנגיש ומשתמש ב-RESTful Service שבניתם בחלק א' של שלב 3.

הקונספט של SPA הינו שבגלישה לאתר אנו מקבלים מהשרת אך ורק דף אחד בשם index.html שרק טוען את ה-CSS ואת ה-JavaScript. מכאן כל בניית הדפים מתרחשת בצד הלקוח ללא גלישה לשום דף אחר. במידה והאתר רוצה להציג מידע או לבצע פעולה אחרת הקשורה לשרת – הוא מבצע גלישת AJAX ל-RESTful Service – דבר שלא גורם לריפרוש האתר, לא טוען דף שלם, לא גורם לאיפוס משתנים בצד הלקוח וכדומה. המידע מהשרת מוחזר מגלישת ה-AJAX בפורמט JSON וצד הלקוח בונה את הדף המבוקש ומציג בו את המידע.

כמובן ש-Angular Framework מקלה על כל זה ומאפשרת לנו לבצע זאת בצורה פשוטה יחסית.

<u>להלן השלבים לביצוע עבור בניית האתר:</u>

1. מתוך ה-Commnad-Line, יש לבצע התקנה חדשה לגמרי של פרויקט אנגולר ע"י הפקודה:
   ng new CouponSystemWebsite
2. יש ליצור Layout מתאים עבור האתר ב-AppComponent או ב-Component אחר המיועד עבור Layout (לדוגמה LayoutComponent).
3. יש ליצור מנגנון Routing מתאים עבור ה-Routs השונים שקיימים בצד הלקוח. על ה-Routes לכלול Default Route המפנה לדף הבית ו-Route עבור Page Not Found.
4. יש ליצור Component עבור דף הבית של האתר המכיל הסבר כללי על האתר ותמונה מתאימה או סרטון הקשור לנושא.
5. יש ליצור Component עבור Login, שבו המשתמש יכניס את הנתונים הבאים:
   - בחירה בסוג ה-Client ע"י תיבת Select שתאפשר למשתמש לבחור באחד מהבאים:
     - Administrator
     - Company
     - Customer
   - הכנסת אימייל
   - הכנסת סיסמה

   בלחיצה על לחצן Login תתבצע גלישה ל-Route המתאים בצד השרת.
   אם פרטי ה-Login נכונים – יוחזר מהשרת ה-Token שבעזרתו המשתמש יוכל להמשיך ולגלוש לדפי האתר הדורשים כניסה. אם פרטי ה-Login אינם נכונים, השרת יחזיר הודעת שגיאה או ערך אחר המציין כי פרטי התחברות שגויים. במצב כזה יש להציג הודעת שגיאה מתאימה ללקוח.

6. לאחר ביצוע Login מוצלח – יש להציג תפריט המתאים לסוג ה-Client:
עבור Client מסוג Administrator – יש להציג תפריט המתאים עבור פעולות ה-Administrator.
עבור Client מסוג Company – יש להציג תפריט המתאים עבור פעולות ה-Company.
עבור Client מסוג Customer – יש להציג תפריט המתאים עבור פעולות ה-Customer.
אין להציג תפריט שאינו מתאים לסוג ה-Client. לדוגמה, אם ה-Client התחבר כ-Customer, אסור לו לראות את התפריט של ה-Administrator או של ה-Company.
7. כל תפריט של Client מכיל את כל הפעולות שאותו ה-Client מסוגל לבצע. לדוגמה, תפריט ה-Administrator יכיל את כל הפעולות שה-Administrator מסוגל לבצע במערכת:
   - הוספת חברה
   - עדכון חברה
   - מחיקת חברה
   - צפייה בכל החברות
   - וכו'

במצב כזה – כל לינק יפנה ל-Component ייעודי בו ניתן יהיה לבצע את הפעולה הדרושה.
ניתן אולם, לאחד מספר פעולות לכדי לינק אחד בתפריט. לדוגמה:
   - הוספה/עדכון/מחיקת חברה

במצב כזה – הלינק של הוספה/עדכון/מחיקת חברה יפנה ל-Component אחד שבו ניתן יהיה לבצע את שלושת הפעולות הללו.
שימו לב: אף לינק לא אמור לבצע Refresh לאתר ולטעון דף שלם, אחרת האתר כבר לא SPA.
8. יש ליצור Component מתאים עבור כל פעולה הקיימת בתפריט.
לדוגמה, Component עבור הוספת חברה, Component עבור עדכון חברה ו-Component עבור מחיקת חברה, אם אלו שלושה תפריטים שונים. במצב כזה כל Component יבצע את הפעולה הספציפית שלו, או Component אחד עבור הוספה/עדכון/מחיקת חברה, אם כל אלו ניתנים ע"י לינק אחד בתפריט. במצב כזה ה-Component הזה יאפשר לבצע את שלושת הפעולות הללו, כמובן ע"י שלושה לחצנים נפרדים – האחד עבור הוספה, השני עבור עדכון והשלישי עבור מחיקה.
9. יש לאפשר ללקוח לבצע Logout מהמערכת. ביצוע Logout כולל גלישה ל-Route המתאים בצד השרת המבצע Logout בשרת וכן מחיקת ה-Token בצד הלקוח. לקוח שביצע Logout מוחזר לדף הבית ולא יכול להיכנס לשאר הדפים ללא ביצוע שוב Login.
אם המשתמש אינו Logged-In, יש להציג לו את לינק ה-Login ואין להציג לו את לינק ה-Logout.
אם המשתמש Logged-In, אין להציג לו את לינק ה-Login ויש להציג לו את לינק ה-Logout.
10. כל גלישה לצד השרת יש לבצע ע"י Custom Angular Service מתאים. לדוגמה, גלישה לשרת עבור פעולות הקשורות ל-Customer יש לבצע ב-CustomerService. גלישה לשרת עבור פעולות הקשורות ל-Company יש לבצע ב-CompanyService וכדומה.

11. יש לבנות מחלקות Models בצד הלקוח המתאימות למחלקות ה-Java Beans שבשרת. לדוגמה עבור קבלת קופון, צד השרת מחזיר אובייקט JSON הנבנה ממחלקת ה-Coupon. בצד הלקוח יש לכן לבנות (ב-TypeScript) מחלקה תואמת בשם Coupon.
12. יש להוסיף ולידציה בצד הלקוח עבור התיבות השונות שבטפסים השונים. לדוגמה, משתמש לא יכול לבצע Login ללא בחירה של ה-Client Type וללא הכנסת אימייל חוקי וסיסמה. אם לדוגמה המשתמש לא הכניס אימייל, הוא יראה הודעה המציינת שחסר אימייל. אם לדוגמה המשתמש מכניס אימייל לא חוקי (חסר @ וכדומה), הוא יראה הודעה המציינת שהאימייל לא חוקי. כמו כן כדאי בנוסף להצגת שגיאות ולידציה, להפוך את לחצן השליחה בטופס ללא פעיל (Disabled) במידה והוולידציה נכשלת.
13. אתגר רשות: חלוקת האתר למספר NgModules (אם בוחרים לבצע זאת, כדאי לבצע זאת ממש מתחילת הפיתוח ולא לבצע את החלוקה רק בסיום הפיתוח).
14. אתגר רשות: ביצוע Lazy-Loading עבור NgModules ספציפיים באתר (רק אם ביצעתם חלוקה ל-NgModules)
15. אתגר רשות: שילוב ארכיטקטורת Redux לצורך שמירת כל המידע במיקום מרכזי אחד.
16. אתגר רשות: שילוב רכיבי Angular Material או Angular Bootstrap להעשרת ה-UI.

דגשים:

- את התמונות השונות אפשר לשמור בתיקייה assets/images בפרויקט ה-Angular.
- חשוב להודיע ללקוח על כל פעולה שביצע – האם הפעולה הצליחה או האם הפעולה נכשלה, ואם נכשלה – להציג הודעת שגיאה מתאימה
- עדיף לא להציג ללקוח קודים של חברות/קופונים וכו'.
  קודים של מסד הנתונים זה לא משהו שהמשתמש אמור לראות.
  וכמובן לא לבקש מהלקוח להכניס קודים של חברות/קופונים בכדי לבצע עדכון או מחיקה, אלא לאפשר לו לבחור מרשימה כלשהי עבור ביצוע עדכון או מחיקה.
- על האתר להיות אסתטי, פשוט וקל לתפעול.
- על המערכת לעבוד ללא שגיאות קומפילציה, ללא באגים וללא קריסות. כדאי להריץ את האתר ע"י דגל ה-Ahead-Of-Time בכדי לגלות יותר שגיאות קומפילציה מהרגיל:
  ביצוע קומפילציה בלבד: ng s --aot
  ביצוע קומפילציה + פתיחת דפדפן: ng s --aot -o

בהצלחה !