**פרויקט הקמת מערכת קופונים (חלק ב') – ריכוז הערות לבודק**

1. חלקיו העיקריים של פרויקט ה-Maven כוללים את ליבת המערכת (כקובץ jar תחת תיקיית lib), קבצי צד שרת (תחת Java Resources) וקבצי צד לקוח (תחת src 🡨 main 🡨 webapp 🡨 view). לצידם ישנו DB. נציין כי קבצי צד שרת נחלקים ל-WebAPIs, ל-Servlets ול-Logger; וקבצי צד לקוח (עבור 3 היוזרים והלוגין) נחלקים לקבצי CSS, לקבצי html ולקבצי script (קונטרולרים, סרביסים, route ועוד).
2. עבור הלוגין, בחרנו ב-id כשדה המזהה עבור חברה/לקוח (במקום שם המשתמש), כך שיוזרים אלה מתחברים עם id וסיסמא, בעוד שאדמיניסטרטור מתחבר עם שם משתמש וסיסמא. לשם הנוחות, צורף לפרויקט קובץ עם פרטי כל החברות והלקוחות, כך שיתאפשר לעשות לוגין בנקל עבור כל יוזר:

<http://localhost:8080/project/view/login/html/login.html>

1. בעת הגלישה למתודות השונות, יש לשים לב שבמספר מקרים עשויה הטבלה לחרוג מגבולות הדף המוצג (כתלות ברוחב המסך שבשימוש). אי לכך, במקרים אלה הוספה גלגלת אופקית, שאותה ניתן למצוא בירידה לתחתית העמוד. המתודות הרלוונטיות הן הבאות: updateCoupon תחת company, וכל המתודות אשר תחת customer (בדגש על purchaseCoupon), למעט readCustomer.
2. כדי שפרטי המשתמש יישמרו גם לאחר שסגר את ה-session, שתלנו cookies עם פרטי ההתחברות. אלה נבדקים בצד לקוח בכל התחברות אל דף הלוגין, ואם נמצאו cookies, מועבר היוזר ישירות אל ה-LoginServlet (מבלי שיהיה עליו למלא את טופס הלוגין). מאידך, אם לא נמצאו cookies, עולה דף הלוגין כרגיל לצורך התחברותו של היוזר למערכת. ב-LoginServlet אנו בודקים אם נשמרו cookies אצל הלקוח. אם כן, מתבצע הלוגין לפיהם, ואם לאו, מתבצע הלוגין עפ"י פרטי הטופס הנוכחי. כך או כך, לאחר לוגין תקין נשמרים ה-cookies על ה-response, והיוזר מנותב לדף המתאים.

קבענו את הגיל המקסימלי של ה-cookies לשבוע, כשניתן כמובן לשנות זאת בנקל לכל גיל מבוקש.

1. לטובת פעולת ה-logout, הוספנו סרבלט ייעודי, אשר מוחק cookies שנשתלו, מסיר את הפאסאד מן הסשן (ע"י invalidation לסשן), ועושה redirect לדף הלוגין.
2. כאשר יוזר מנסה לגלוש לאתר או לאחת המתודות ללא לוגין, הוא מנותב לצפייה בהודעת שגיאה של 401 (הניתוב מבוצע, כמובן, בהסתמך על האקספשן אשר נזרק מה-WebService). בחרנו שההודעה תוצג ברמת הדף, מבלי שהיוזר יופנה לדף שגיאה חיצוני, שכן בצורה זו הוא יכול בקלות ללחוץ על כפתור שמפנה אותו לדף הלוגין (בדף השגיאה שמוצג לו מופיע מלל מתאים בנושא).

להתחברות לדף הלוגין, כבסעיף ב' (לאמור, התחברות תקינה לאתר הרשמי של מערכת הקופונים):

<http://localhost:8080/project/view/login/html/login.html>

לעקיפת הלוגין דרך האתר של אדמין, למשל:

<http://localhost:8080/project/view/admin/html/admin.html>

לעקיפת הלוגין דרך אחת המתודות, למשל:

<http://localhost:8080/project/view/admin/html/admin.html#/readCompany>

לקבלת האקספשן ישירות מהשרת דרך ה-path שמוגדר ב-WebService, למשל:

<http://localhost:8080/project/webapi/admin/companies>

חשוב לציין כי השתילה של cookies בעת הלוגין הראשוני מאפשרת ליוזר לגלוש בחופשיות רק ללינק האתר הרשמי (הראשון הנ"ל). מאידך, בניסיון לגלוש ישירות לאתר היוזר או לאחת המתודות, יופיע דף השגיאה. במקרה כזה, יופנה היוזר לכפתור ה-Switch user/Logout אשר בדף היוזר, כך שינותב לעמוד הלוגין לאחר שעבר דרך ה-LogoutServlet, תוך מחיקת ה-cookies כמובן. בצורה זו מובטחת גלישה מסודרת ללינק המייצג של האתר, גם אם מדובר ביוזר שעשה בעבר לוגין תקין!

1. שינינו את מאגר הנתונים המשמש אותנו במסגרת הפרויקט מ-Apache Derbey ל-MySQL.
2. בפרויקט יישמנו את ה-Richardson Maturity Model, על שני שלביו הראשונים. על-פי המודל, יישום REST מתחלק לשלושה שלבים, כאשר הטמעת שלושתם הופכת אותו ל-Restful.
3. כל המתודות ב-WebAPIs הותאמו להחזרת Response, כש-Response תקין נכתב בפורמט קבוע של סטטוס, location (מעביר את ה-URI של המתודה ב-header) ו-entity (האובייקט עצמו שמוחזר). מאידך גיסא, במקרה של Exceptions, לא מצאנו טעם בהחזרת ה-location במסגרת ה-Response.
4. הסטטוס המוחזר ב-Response של כל Exception נקבע בהתאם לאופי ה-CouponSystemException שעלול להיזרק מן ה-Facade של חלק א' של הפרויקט. לדוגמה, במתודות ה-POST מדובר בבקשות לא תקינות מצדו של היוזר, כגון הזנת ערך ריק באחד השדות, ולפיכך נקבע סטטוס של 400 (Bad request). כמו כן, כל מתודה פונה תחילה למתודה פנימית של ה-class בשם getUserFacade, אשר במסגרתה נזרק LoginException אם אין פאסאד על הסשן. על כן, בכל מתודה קיים גם catch-block אשר מטפל באופן ספציפי במקרה של LoginException, ואשר מחזיר Response עם סטטוס של 401 (Unauthorized). בנוסף, למקרה שנזרק Exception כללי יותר, הוספנו catch-block כללי, שמטפל בכל אותם Exceptionים אפשריים, ומטבע הדברים הותאם עבורו Response בעל סטטוס כללי של 500 (Internal server error).
5. במתודות שמחזירות בוליאן מהפאסאד, ייתכן false גם מבלי שייזרק Exception. לפיכך הפרדנו את ה-Response התקין (לכאורה) לשני מקרים, כך שסה"כ הבחנו בין ארבעה סוגי Response אפשריים: Response עבור CouponSystemException, Response עבור Exception כללי, Response עבור מתודה שהסתיימה בהצלחה (true), ו-Response עבור מתודה שהסתיימה בכישלון (false). ניסינו, כאמור, להתאים כל סטטוס לאופי המתודה.
6. בהמשך להערות הנ"ל, מן ראוי לציין כי הוספנו ולידציה ברמת ה-UI, בשאיפה להימנע מזריקה של Exceptions ככל הניתן. רק במקרים בהם לא ניתן למנוע את הדברים ברמת הדף, תיתכן זריקה של Exception, לדוגמה כאשר מנסים ליצור חברה עם שם שכבר קיים במערכת (ואז הבדיקה חייבת להיעשות מול ה-database). כמו כן, ה-Exceptions מהווים רשת ביטחון למקרה שהולידציה כושלת.
7. מכיוון שרוב המתודות מקבלות pathparam בנוסף לאובייקט JSON, וידאנו במקרים הרלוונטיים כי הנתונים המתקבלים בשני האופנים תואמים. אזי, רק במקרה של התאמה בין הנתונים, חלה קריאה למתודה שב-Facade. יתר על כן, במתודות PUT, לדוגמה, שקוראות למתודות המחזירות בוליאן מה-Facade (אשר משקף את הצלחת הפעולה), ייתכן סטטוס false על אף שלא נזרק Exception. אמנם יצרנו צד לקוח, אשר בפועל איננו מאפשר חוסר התאמה, אולם ה-Service בצד שרת נדרש לתמוך באפשרות זו של חוסר התאמה, שהרי באופן כללי לא נוכל להסתמך על צד לקוח לשם כך (בפידלר, למשל, עדיין ניתן לשלוח JSON שלא יתאם את ה-pathparam).
8. יצרנו Logger ב-class נפרד (סינגלטון מטבעו), וכתבנו אליו בכל הקבצים של צד שרת (שלושת ה-WebServices ושני הסרבלטים). ה-Logger עושה שימוש ב-FileHandler וב-SimpleFormatter (אשר מוגדרים ב-CTOR) לשם כתיבה לקובץ (נוצר ב-C:\eclipse, או בהתאם למיקום תיקיית ה-eclipse).

הכניסה לכל מתודה, הצגת הפרמטרים אשר נמסרו למתודה, פעולות במתודה והיציאה מהמתודה הוגדרו כולם ב-level של INFO, בשעה שכל האקספשנים שנתפסו הוגדרו ב-level של SEVERE, תוך הדפסת מחרוזת מפורטת של האקספשן שנתפס (e.toString()). בנוסף, במתודות העדכון והמחיקה המוגדרות בסרביסים, שעלולות להסתיים עם סטטוס false שלא כתוצאה מאקספשן (כאמור בהערות קודמות), הגדרנו את היציאה מהמתודה במקרה שכזה ב-level של WARNING.

חשוב לציין שבמסגרת הכתיבה של הלוגר לקובץ עלולים להיזרק אקספשנים. הטיפול, במקרה כזה, יכול להיות אחד משניים: try-catch בעת הגדרת הכתיבה לקובץ ב-CTOR של הלוגר, כך שלא יהיה לוגר שכותב לקובץ במקרה של אקספשן (ונסיק שהייתה בעיה עקב חוסר התיעוד בקובץ); או הוספת הצהרת throws, אותה יש לזרוק הלאה גם בכל קריאה אל הלוגר, בכוונה תחילה שלא יהיה טיפול באקספשן, כך שהמערכת תיתקע בהיעדר לוגר שכותב לקובץ. שני הטיפולים קבילים, כמובן, ותלויים בדרישת הלקוח. אנו יישמנו את האפשרות הראשונה, אך באותה מידה יכולנו ליישם את זו השנייה.

1. ניתן למצוא הערות תיעוד עבור כל המתודות שבצד שרת (הן בסרביסים והן בסרבלטים) וכן הערות תיעוד בצד לקוח.
2. עשינו שימוש בספריה Angular Confirm (מושתתת על bootstrap ועל J-query) ע"י dependency injection . הספריה מאפשרת הצגת Confirm מותאם אישית.
3. עשינו שימוש ב-ngAnimate ע"י dependency injection. הספריה שימשה לביצוע מעבר חלק בין הדפים השונים ב-SPA וכן ברענון.
4. בוצע שימוש ב-CSS3 לצורך עיצוב קצוות תמונות הלוגו. בנוסף, לצורך ה-Exceptions, נבנה pop-up (קובץ exceptionMessage.css), אשר נכנס ל-6 שניות להצגת ההודעה, ולאחר מכן יוצא, ומבוצע רענון אוטומטי. לצורך הדגמה, ניתן במתודת יצירת חברה להכניס שם חברה קיים ותתקבל ההודעה.
5. בהמשך לאמור לעיל, במסגרת ולידציה בוצע שימוש ב-ng-pattern לצורך מתן דפוס לולידציה. ניתן לראות דוגמה לכך במתודת עדכון חברה בדף האדמין. כהרחבה ל-type=”email” – אשר אינו דורש נוסח מלא של מייל, אלא רק קיומו של @ – כאן נדרש נוסח מלא, הכולל דומיין תקין, כאשר סיומת הדומיין חייבת להיות בעלת 2 אותיות לפחות (כנדרש ממייל תקין וסטנדרטי).

כפתור ה-update/submit רגיש למילוי תקין של שדות ה-input, ויהיה זמין ללחיצה רק במקרה שכל הפרטים שהוכנסו תקינים בהתאם לולידציה. במתודות העדכון ישנה אפשרות לעדכן רק רשומה אחת כל פעם. הדבר שומר על הזנת המידע ועל UX תקין ונוח. הולידציה בוצעה ברמת דף ה-html וגם ע"י שילוב מתודות ב-controllerים. חל שימוש במגוון כלי ולידציה הקיימים ב-angular.

1. לצורך הצגת הקופונים שלא נרכשו בדף הלקוח, נדרשה הרחבה לחלק א' של הפרויקט, במסגרתה יצרנו מתודה חדשה בליבת המערכת, ושילבנו אותה בצד הלקוח.
2. באופן כללי, עשינו שימוש נרחב ב-bootstrap בעיצוב טבלאות, כפתורים ורספונסיביות.
3. בדף הלוגין ישנה הצגה של תמונות באמצעות bootstrap carousel. התמונות ניתנות להחלפה ע"י אינדיקטורים, מקשי חיצים בכל צד, ובאופן אוטומטי. התמונות המוצגות הן חלק ממערך, אשר קיימות בו יותר משלוש התמונות המוצגות, ובכל רענון נבחרות שלוש תמונות באופן אקראי. לצורך הכנסת התמונות למערך בוצע שימוש ב-custom directive, שאותו ניתן לראות בקובץ login.html באמצעות התגית <yl-coupons-images mycoupons=*"mycoupons"*></yl-coupons-images>. לשם כך בוצע שימוש ב-isolated scope של two way binding ("=").
4. השתמשנו ב-directive של ng-options עבור המתודות של קריאת חברה/לקוח/קפון, אשר במסגרתן הצגנו את הרשומות הניתנות לקריאה כרשימה סגורה, שכן זו מספקת ליוזר שימוש קל ונוח במתודה.
5. עשינו שימוש ב-HTML5 בדף של ה-customer: ניתן לראות וידיאו בתנועה בזכות השימוש בתגיות <video> ו-<canvas>. כמו כן, לצורך עצירת ניגון הוידיאו, ניתן ללחוץ קליק אחד, אשר מביא להפעלת המתודה video.pause(). לצורך הפעלה הוידיאו מחדש, יש ללחוץ דאבל קליק, ובכך תופעל המתודה video.play().
6. הצגת התמונות של הקופונים באמצעות base64: במסגרת יצירת קופון בדף חברה, נשמרת התמונה ב-database בקידוד של base64. את העלאת התמונה ניתן לבצע ע"י drag n drop או ע"י לחיצה על אזור הגרירה, כך שייפתח Browse לצורך בחירת הקובץ להעלאה. ניתן לשמור עד גודל של 16 מגה לתמונה. לצורך כך נעשה שימוש ב-MEDIUMBLOB בשדה IMAGE של קופון ב-database.
7. בהמשך להערה קודמת, שינינו בליבת המערכת את ה-toString של קופון (במחלקת ה-JavaBean), כך **שלא** יציג את שדה התמונה במלואו, אלא מלל אשר מציין שהתמונה עברה קידוד ל-base64. זאת מכיוון שאין למחרוזת המוצגת משמעות ממשית, ועל אחת כמה וכמה בהיותה כה ארוכה (תעמיס על קובץ הלוגר שורות רבות, ובכך תקשה על קריאתו בעת הצורך). במקרה הצורך, ניתן תמיד לפנות אל ה-database באופן ישיר לקבלת המחרוזת, שמייצגת את התמונה.
8. ניתן לראות שימוש ב-Jquery וב-CSS3 בדף המתודה readCompany תחת היוזר של company. ישנו עורך תמונות ב-Jquery, וכמו כן, התמונה נערכה עם מספר מאפיינים של CSS3. בין היתר, כשעוברים עם סמן העכבר על התמונה (hover), התמונה גדלה.
9. חל שימוש ב-auto increment לצורך יצירה של id ב-database, ולכן, בעת יצירת חברה/לקוח/קופון, שדה ה-id הינו מוסתר (hidden). ה-id נשלח בתור 0, ומקבל את ערכו האמיתי ב-database באופן אוטומטי עם התווספותה של הרשומה אל מאגר הנתונים. בכך נשמר מספור אוטומטי, מדורג ותקין.
10. במתודה של createCoupon שבדף חברה, ניתן להשתמש ב-Google vision web API בעת טעינת תמונה. זהו REST API, הבנוי באמצעות מודלים של machine learning. הוא מאפשר להוציא מידע על התמונה במגוון קטגוריות. לצורך שימוש מיטבי, יש לבחור קטגוריה **טרם** העלאת תמונה (ובמידה שלא תיבחר קטגוריה ע"י היוזר, ברירת המחדל תהיה LABEL\_DETECTION). לאחר טעינת התמונה, יופיע המידע בפורמט של JSON. לצורך השימוש ב-web API, נדרשנו להשתמש ב-API key, שאותו קיבלנו באמצעות Google cloud. בכל בקשה נשלח ה-API key יחד עם התמונה ב-base64, ומוחזר המידע בהתאם לקטגוריה הנבחרת.

לשם הנוחות והסיכום, מפורטים להלן מרכיבי הפרויקט כולו, החל מדרישות הבסיס, המשך בפיצ'רים שהוספו ב-level הנמוך יותר (כל הארבעה הנדרשים), וכלה בפיצ'רים שהוספו ב-level הגבוה יותר (תשעה במספר):

* Coupon System Core (part A of the project)
* WebAPIs (Admin, Company and Customer)
* Angular Services (Admin, Company and Customer)
* Ui-route (Admin, Company and Customer)
* Login page
* Documentation (in both server side and client side)
* Angular custom directive
* Form validation
* CSS3
* Logger
* Response
* Bootstrap
* Bootstrap-carousel
* HTML5
* MySQL
* Cookies
* J-Query
* Base64 encoding (for coupon images)
* Google vision web API (machine learning)