מטלה 3 / שאלות פתוחות - קבוצה 7

חברי הצוות

שם מלא: היא קבלאן / ת"ז: / מייל: @gmail.com

שם מלא: שלומי לוי / ת"ז: / מייל: @gmail.com

שם מלא: אריק סקיגין / ת"ז: / מייל: @gmail.com

שם מלא: סמי עודה / ת"ז: / מייל: @gmail.com

שם מלא: אור אליהו / ת"ז: / מייל: @gmail.com

שאלות

1. תארו את תהליך התכן (design) שביצעתם באופן הבא:
2. תארו דילמות הנדסיות (Design issue) ספציפיות (לא כלליות) בהן עסקתם בתכן של ביצוע חיפוש מפות. תארו את השיקולים וקבלת ההחלטות שלכם והסבירו את הפתרונות שבחרתם.
3. ציינו איזו מהנושאים מההרצאות 4-9 בקורס באו לידי ביטוי בתהליך התכן המערכתי שביצעתם והסבירו באיזה אופן – באמצעות דוגמאות ספציפיות ישירות מתוך המערכת "GCM".
4. תארו את תהליכי הבדיקות השונים שביצעתם במהלך פיתוח הפרויקט שלכם. ציינו את מאפייני תהליכי הבדיקות תוך התייחסות לעקרונות שנלמדו בהרצאות, ותוך מתן דוגמאות ספציפיות שביצעתם (או לא ביצעתם) במהלך הפרויקט (ע"י תיאור מפורט של בדיקות מרכיבים ספציפיים של מערכת "GCM").
5. תחקור והפקת לקחים: התייחסו לאופן שבו התנהלתם לגבי 2 מרכיבים של ביצוע הפרויקט:
6. תיאום פעילויות ושיתוף בין חברי הצוות בפיתוח וגישה לניהול גרסאות: תארו את השיטה שלפיה פעלתם בהקשרים אלה, וציינו יתרונות וחסרונות שלה. יש להתייחס גם לתהליך העבודה – לא להתמקד רק בכלים ואספקטים טכניים.
7. שילוב קוד (אינטגרציה – לאחר הפיתוח הראשוני) ובדיקות. ציינו באופן פרטני, בהתייחס ספציפית לפיתוח המערכת ""GCM, אך פעלתם בשלב זה של הפיתוח (משל: תיאור התנהלות התהליך, אופן טיפול בבעיות, וכו'). אם היו קשיים מה הסיבה לכך? מה הייתם משנים בדיעבד בגישתכם למרכיב זה מבחינת האספקטים הרלבנטיים של הנדסת תוכנה?

תשובות

1. במימוש ביצוע חיפוש מפות נפגשנו עם מספר דילמות הנדסיות (Design issue):

* יצירת חלון חיפוש וסוגי חיפוש - בממשק שלנו מימשנו בכל חלון עם הרבה מידע חיפוש בצד, חיפוש זה הוא חיפוש "יבש" ז"א בלי התקשרות למול מסד הנתונים,

כאשר הגענו לשלב שצריך לענות על דרישת החיפוש היותר מעמיקה עלתה לנו דילמה האם באותו חיפוש לאפשר חיפושים שונים ז"א גם עבור האלמנטים במסך וגם עבור הדרישות.

בתוך החיפוש המעמיק קיימות אפשרויות שונות שלא בהכרח יענו על הקריטריון הזה לדוגמה: חיפוש לפי שם עיר, לפי נקודת עניין, לפי תיאור מילולי.

לכן, כדי להקל על הממשק שכן בפרויקט שלנו כולל אלמנטים רבים וכדי לאפשר נוחות מקסימלית ע"פ קורס ממשק אדם מחשב החלטנו להפריד את החיפוש כחיפוש מתקדם שבו המשתמש יבחר את סוג החיפוש ומה הוא ירצה לחפש.

* ביצוע החיפוש – כפי שתיארנו למעלה החלטנו כי עבור מסך החלון חיפוש המתקדם הצגנו למשתמש 4 אפשרויות חיפוש מתקדמות,

הדילמה הראשונה שלנו הייתה כי נאחד את החיפוש היבש עם החיפוש למול המסד נתונים אך זה העלה המון בעיות מימוש פוטנציאליות והחלטנו לוותר על זה,

לאחר מכן אחרי שעלה הרעיון של חלון החיפוש, החלטנו אולי בחלון זה לאחד את ה4 חיפושים לתבנית אחת של חיפוש, אך דבר זה ייצור אי שליטה למשתמש בחיפוש ז"א בתוצאות החיפוש הוא לא יידע מה בדיוק עונה על קריטריון החיפוש בכל תוצאה בנוסף לכך נכנס שוב שיקול של צד המשתמש שכן הוא לקוח פוטנציאלי והוא אמור לרכוש ע"פ אותם חיפושים שהוא מבצע.

התלבטות זאת באה ליידי ביטוי גם בשאילתה מכיוון שנצטרך לאחד 4 שאילתות חיפוש שכן דבר זה יכול להכביד על מסד הנתונים בחיפוש.

בסופו של דבר הוחלט כי נעשה הפרדה מוחלטת הן של השאילתות הן של ממשק המשתמש כדי שלמשתמש יהיה שליטה בחיפוש, ולמסד הנתונים יהיה שאילתות יותר קלות לבצע.

* טקסט החיפוש – כחלק מההתלבטויות הכלליות שלנו עלה רעיון של אחד מחברי הצוות כי כלל החיפושים יהיו נטולי case sensitive כדי לתקן בלבול של המשתמש שכן המקום museum הוא בדיוק כמו Museum

בנוסף כדי להגדיל את היצע החיפוש כתבנו את השאילתות בצורה של הכלת הטקסט

ז"א במידה ויבצע חיפוש של museum הוא ימצא גם את x museum ו-y museum.

1. הנושאים שנלמדו בהרצאות בקורס באו לידי ביטוח בתהליך התכן המערכתי שביצענו:

* Reuse – זהו העיקרון המתייחס לשימוש חוזר של קוד, עיקרון זה הוא חלק מהעקרונות ששמנו עליהם דגש מרכזי,

מכיוון שסדר הגודל של הפרויקט הוא גדול מאוד רצינו לצמצם ביצוע קוד חוזר, להשתמש בספריות פתוחות שיעזרו לנו לשב קטעי קוד ומרכיבים אשר לא אנו תכננו אך עונה על דרישות התכנון שלנו וכו'

כחלק מעיקרון זה השתמשנו ב-OSCF המבצע קשר בין ה-server ל-client ובו העברנו מידע הכולל ערכים, פקודות, שגיאות וכו'.

בנוסף השתמשנו בספריית Jackson ע"מ לבצע העברת נתונים בשימוש בפורט הטקסטואלי המוכר json, הכולל המון יתרונות מרכזיים, קריא לאדם (ז"א בביצוע debugging לקוד ראינו את המידע שהאובייקט העביר, דבר שהקל בצורה משמעותית על כתיבת הקוד ומציאת תקלות).

עוד שימוש בספריות הוא עבור ממשק המשתמש שבו השתמשנו בספרייתanimation fx כדי לבצע אפקטי מעבר בחלונות.

בנוסף חוץ משימוש בספריות חיצוניות (framework) כדי לחסוך בכתיבת קוד חוזר בחלון הקטלוג ובחלון הedit catalog עבור המשתמש והעובד השתמשנו באותו controller המשנה את מצבו בהתאם להרשאה למרות שמדובר בשני כפתורים שונים כדי להיכנס למסכים אלו.

* מודל שלוש השכבות – את הקוד תוכנית חילקנו ל3 חלקים (בלי קשר למודל 3 שכבות), client, server, common בחלק המשותף הוספנו את ה-entities של המערכת, כל המחלקות האלה מוכרות ע"י צד הקליינט וצד הסרבר (בגלל אופן התכנון שלנו העברו את המידע ע"י json לכן תכננו מחלקה אבסטרקטית בשם AbstractJsonToString שכולן יורשות אותה ע"מ להוסיף את התכונות הנדרשות להמרה ל-json),

בצד הקליינט יש לנו את guiControllers המסווגים בתכנון כ-boundary,

בנוסף בכל צד הקליינט קיימים לנו requestHandler ו- responseHandlerהמסווגים בתכנון כ-controllers והם משמשים ע"מ לקשר את רכיבי הגוי לבקשות ולתשובות מול הסרבר, ז"א שהממשק מבקש נתונים הוא פונה ל- requestHandlerשלו, ב- requestHandler מבוצע עיבוד המידע הנדרש מהסרבר ושולח את הבקשה דרך ה-ocsf של הקליינט, כשהמידע חוזר, תחילה מגיע ל-ocsf משמה נשלח לעיבוד ב-responseHandler ואחרי העיבוד נשלח לגוי.

* אינטגרציה – כפי שהוסבר בהרצאה, מפאת חוסר ידע טכני וזמן מצומצם לא יכלנו להשתמש באינטגרציה רציפה מה שכן שזאת בין הדרכים הנוחות שמשתמשים בהן היום בתעשייה בשילוב עם מערכת גרסאות (גיט).

אבל כן השתמשנו במטרת האינטגרציה ע"מ לשלב בין חלקי התוכנה לאורך כל דרך המימוש, ז"א הגדרנו בקבוצה פונקציונליות של המערכת הכוללת את צד הסרבר והקליינט, בסיום עבודת הצוות דאגנו לבצע איחוד ובדיקות לפונקציונליות החדשה ולראות שהיא עובדת כפי שתכננו אותה.

1. במהלך פיתוח הפרויקט ביצענו בדיקות שונות שמטרתן הייתה בדיקת נכונות התוכנית מבחינת קוד, פונקציונליות וממשק.

תחילה חילקנו את חברי הקבוצה ל2 צוותים, קליינט וסרבר, כאשר כל צוות מתנהל בפני עצמו כדי למקסם את העבודה ואת הידע הנצבר עבור החלקי הספציפי שהם מפתחים,

כאשר מישהו מחברי הצוות כותב קוד ובודק אותו, חבר צוות העובד אתו במשותף בודק את הקוד שבדק וזאת ע"מ לוודא שקטעי הקוד נבדקים בצורה אופטימלית ללא עיכובים מיותרים שימנעו באגים בעתיד ככל שאפשר.

בנוסף כל צוות אף הגדיל וקרא באינטרנט על גישות נוספות של מימוש התכנון בצורה אופטימלית ע"מ לקצר תהליכים בזמן הבנייה והבדיקות.

בנוסף כאשר הרמנו את חלק התקשורת מצד הקליינט ומבחינת הסרבר נציג משני הצוותים מבצעים בדיקות כדי לראות שהחיבור מתבצע בצורה אופטימלית ועונה על הדרישות התכנוניות שביצענו.

לדוגמה, כאשר סיימנו את חלק ההתחברות בצד הקליינט והסרבר ישבנו שני הנציגים משני חלקי הפרויקט כדי לראות האם מה שנשלח זה מה שאמור להתקבל והפוך(אינטגרציה),

בנוסף כחלק מדרישות הפרויקט שרק משתמש אחד אמור להיות מחובר למערכת, הם ביצעו בדיקות כדי לראות שרק משתמש אחד יוכל להיות במערכת וכאשר משתמש מבצע התחברות שנייה ממחשב אחר, ההתחברות מהמחשב הראשון מתנתק ומקבל שגיאה בהתאם.

כאשר צוות הקליינט סיים את חלוקת המסכים כולל ההרשאות, הצטרף אחד נציג מצוות הסרבר כדי לראות שההרשאות מתקבלות כראוי וצוות הקליינט בדק כי אחד השינויים מבוצעים על ממשק המשתמש.

בנוסף היו בדיקות שלא היינו צריכים נציגים משני הצוותים, ובוצע בדיקה פנימית של כל חברי הצוות, לדוגמה צוות הסרבר היה צריך לבדוק כי החוט שרץ ע"מ לשלוח הודעות על פקיעת התוקף 3 ימים מראש. לכן צוות זה ביצע "הרצה בכוח" על החוט ובדק במסד נתונים כי הערכים נכנסים כראוי.

דוגמה נוספת עבור צוות הקליינט היא בסיום בניית הממשק, צוות זה ערך בדיקות של תקינות קלט, כפתורים, עומס נתונים וכו'.

בשלב הסופי (כיומיים לפני ההגשה) כל חברי הקבוצה ישבנו וכתבו תרחישים העולים להפיל את התוכנה שבנינו מצד הסרבר ומצד הקליינט.

ואכן מצאנו הרבה כאילו דבר שסגר לנו באגים רבים בתוכנית.

דוגמה לכך שחבר צוות העלה הוא כאשר בשדה מסוים נכניס מספר תווים הגדול יותר משהוגדר במסד הנתונים או לחילופין שדות ריקים שלא אמורים להיות.

כדי לבצע את הבדיקות בצורה אופטימלית וכפי שלמדנו בקורס מבוא לבדיקות תוכנה הוספנו בדיקות יחידה (Junit) על חלקי פרויקט מרכזיים ע"פ האפשרויות של

* "קופסה שחורה" – בדיקת תפקוד רכיב במערכת, בדיקת נכונות פלט, עיקרי הבדיקה היא עבור תהליכים מורכבים לדוגמה מעבר בין מסכים שדרכו עוברות גם הקלטיים ממסך למסך.

לדוגמה במסך הרכישה, אם לקוח בוחר בעיר שעדיין לא רכש אותה הוא מקבל הודעה בpopup שהוא לא רכש אותה והאם מעוניין לרכש, אם הלקוח לוחץ כן אז מעביר אותו למסך הרכישה.

דוגמה נוספת עבור הוספת מקום במקום, המשתמש לוחץ על מקום שברצונות להוסיף מקום, כאשר לחץ נפתח לו חלון עם כל המקומות בעיר, כאשר בחר מקום נסגר החלון הנוסף ועל המפה יתווסף המקום שבחר.

* "קופסה לבנה" – בדיקות תפקוד המבנה הפנימי של רכיבים ע"י השמת דגש עבור המעברים הפנימיים (מסלולי החישוב) ובדיקות פונקציונליות מורכבות.

לדוגמה: במסך של הקטלוג/עריכת קטלוג השתמשנו באותו controller המשנה את מצבו בהתאם להרשאה או לפונקציונליות הנדרשת.

אזי מכיוון שפונקציונליות זאתי מורכבת מאוד הכוללת בדיקת זליגת הרשאות או קבלת הרשאות נצטרך לבדוק תרחישים קיצוניים היכולים לגרום לאי נכונות.

לדוגמה בעריכת קטלוג למנהלת מחלקת תוכן ולמנהלת חברה קיימת הרשאה נוספת של מחיקה מה שלא קיים עבור עובד מחלקת תוכן.

(א). בגלל מורכבות הפרויקט מבחינת היקפו וזמנו הקצר כ-3 חודש בלבד הכולל תכנון ופיתוח עוד בתחילת העבודה הוחלט על פתיחת קבוצת WhatsApp שבה עדכנו באופן שוטף על הקורס ועל התקדמות העבודה.

עם קבלת מטלה 3 – מימוש התכנון התחלקנו לשני צוותים:

* צוות 1: פיתוח צד הקליינט
* צוות 2: פיתוח צד הסרבר ומימוש החלק המשותף

בכך מקסמנו את יכולת העבודה במקביל של הפרויקט, בנוסף במידה והתאפשר הצוותים התחלקו לתתי צוותים.

לצוותים הגדרנו גבולות ברורים שבניהם במידה והם מצאו בעיה בצד של הצוות השני אסור להם לגעת בקוד, אלא רק לדווח להם.

בשלב חלוקה לצוותים ידענו כי בצד הסרבר יהיו רוב בעיות מכיוון שצד זה כולל את החלק המשותף, התקשורת למול הDB, בניית השאילתות והתקשורת למול קליינט, לכן דאגנו שבצד יהיה 3 חבריי צוות.

בשלב האבטיפוס דאגנו כי נעבוד בצורה הכי טובה שאפשר בצמוד לתכנון ע"מ שהוא יהיה הבסיס לפיתוח הפרויקט, ואכן דבר זה עזר להמשך הפרויקט, בנוסף כבר בשלב זה פתחנו GIT ע"מ לסנכרן את הקוד של הצוותים.

במקביל בכל שלב בפרויקט דאגנו לבצע גיבוי ידני (כ-3 פעמיים ביום עבודה!) למול הGIT כדי שמידה ויקרה משהו קיצוני תמיד יהיה לנו גיבוי לשחזור, למזלנו לא היינו צריכים.

היתרון המרכזי של שיטת העבודה שלנו הוא שלמרות שעבדנו במקביל דאגנו תמיד לדווח בתוך הצוותים על השינויים שאנחנו מבצעים דבר שבא לידי ביטוי בכך שהיו לנו מקרי קונפליקטים בודדים בלבד!

בגלל שהשתמשנו ב-GIT לניהול גרסאות במידה ועשינו שינוי שעשה בעיות בהמשך הפיתוח היינו בודקים ברמת הקובץ מתי שונה והיכן ומשחזרים את חלקי הקוד.

יתרון נוסף הוא שבלחיצת כפתור כל חבר צוות יכול לקבל את הגרסה האחרונה של הפרויקט ובכך יכול להבחין בבאגים שהוא נתקל בהם ולעדכן את חברי הקבצה על כך.

החסרונות של השיטה שעבדנו בה היא שבמידה ואינטגרציה של פונקציונליות מסוימת התעכבה בגלל סיבות שונות שאר חברי הקבוצה מנועים מלעשות push וpull (בגלל שלא יקבלו פונקציונליות שלא נבדקה), בדיעבד דבר זה יכול היה להיפתר במידה והיינו עובדים עם הענפים בGIT.

(ב). כפי שכתבנו בסעיף א', את הבסיס לפרויקט כבר הכנו בשלב האבטיפוס, בסיס זה הכיל ברובו את יחידות התקשורת הכמעט מוגמרות.

בתחילת המטלה הקפדנו כי תחילת העבודה למרות החלוקה לצוותים תהייה פיזית ביחד ע"מ לתאם את אופן העבודה בצורה הטובה ביותר, לאחר שלב זה הצוותים התנהלו בפני עצמם ובסנכרון בקבוצת הWhatsApp (לקראת סוף המטלה עבדנו גם באופן זהה).

בבניית הבסיס לפרויקט מנענו אינטגרציה של חלקים מרכזיים וגדולים מאוד בקוד ובעצם כך תשית התקשורת הייתה מונחת בין הקליינט לסרבר דבר שגרם לעבודה כמעט ללא תלות של הצוותים.

דוגמה לתיאור תהליך אינטגרציה:

כאשר אחד מחברי הקבוצה סיים חלק מסוים הוא מבצע push ומעדכן בקבוצת ה-WhatsApp כאשר אחד מחברי הצוות השני קיבל את ההודעה הוא מצטוות אל אותו חבר קבוצה ורק הוא עושה pull, לאחר מכן, שניהם מבצעים בדיקות ומריצים את התוכנית במצב debugging mode (בשביל מצב זה יצרנו מצב נסתר בקליינט שבהרצת התוכנית עם ארגומנט ראשון debuggingMode חלון זה יפעל), דוגמאות ספציפיות הראנו בשאלה 2. לאחר שבדקו שהכל עובד כראוי שאר חברי הצוות מבצעים pull.

בדיעבד היינו משנים את תחילת העבודה שלנו עם מערכת ניהול הגרסאות GIT שכן היא חדשה לנו, וכל אחד קרא עליה בנפרד באינטרנט וכל חבר קבוצה העמיק בלמידה בצורה שונה, דבר שבא ליידי ביטוי בהמון שאלות של חברי קבוצה אחרים, ודרש הסברים. לכן כחלק מהמפגשים הראשונים היינו צריכים לתת לזה יותר דגש בהסבר המערכת.

בנוסף לכך היינו משנים את תכנון הזמן הכולל של הפרויקט שכן לא לקחנו בחשבון את בדיקת המוצר המוגמר, שמו לב כי בבדיקת המוצר הסופי היו באגים ושינויים קוסמטיים שהינו חייבים לבצע, דבר שלא מומלץ לעשות כאשר זה נעשה בלחץ זמן.