תוכן

[1. מבוא 1](#_Toc1063595)

[1.1 הגדרת הבעיה 1](#_Toc1063596)

[1.2 ואיך הפרויקט פותר אותה 1](#_Toc1063597)

[1.3 במידה ויש לו מתחרים 1](#_Toc1063598)

[1.4 אולי סיכום קטן על הפרק 1](#_Toc1063599)

[2. רקע תיאורטי 3](#_Toc1063600)

[2.1 רשתות מחשבים 3](#_Toc1063601)

[2.2 מודל client – server 4](#_Toc1063602)

[2.3 ניתוח כלי פיתוח 5](#_Toc1063603)

[2.3.1 סביבת עבודה 5](#_Toc1063604)

[2.3.2 שפות פיתוח 9](#_Toc1063605)

[2.4 מודל הפיתוח 14](#_Toc1063606)

[3. תכנון 14](#_Toc1063607)

[3.1 דרישות המערכת 14](#_Toc1063608)

[3.1.1 דרישות כלליות 14](#_Toc1063609)

[3.1.2 דרישות side client 14](#_Toc1063610)

[3.1.3 דרישות side server 14](#_Toc1063611)

[3.2 אפיון המערכת 14](#_Toc1063612)

[3.3 מבנה המערכת (אופציונאלי – לא בטוח שזה יופיע בדווקא כאן) 14](#_Toc1063613)

[4. תכנון ביצוע 15](#_Toc1063614)

[4.1 תחילת העבודה 15](#_Toc1063615)

[4.2 מודל הפיתוח של המערכת (כאן כדאי להציג מודלים – כדאי לעיין גם בפרויקט של חירום בפיקוד העורף ) 15](#_Toc1063616)

[UI 4.3 15](#_Toc1063617)

[4.4 הקמת שרת Node.js 16](#_Toc1063618)

[4.5 הקמת instance in AWS 16](#_Toc1063619)

[4.6 ארגון בסיס הנתונים (אולי צריך לשלבו עם 4.5) 16](#_Toc1063620)

[Data Mining4.7 (אולי גם את זה נוסיף פשוט ל4.5) 16](#_Toc1063621)

[4.8 תהליכים 16](#_Toc1063622)

[4.9 פונקציונאליות 22](#_Toc1063623)

[4.10 בדיקות 22](#_Toc1063624)

[5. דיון 22](#_Toc1063625)

[5.1 יתרונות האפליקציה 22](#_Toc1063626)

[5.2 הרחבות אפשריות למערכת 22](#_Toc1063627)

[5.3 רווחים אישיים מהפרויקט 22](#_Toc1063628)

[6. סיכום 22](#_Toc1063629)

[7. ביביליוגרפיה 22](#_Toc1063630)

# מבוא

## 1.1 הגדרת הבעיה

## 1.2 ואיך הפרויקט פותר אותה

(האם פותר באופן חלקי? מה היינו מוסיפים? האם פותר באופן מלא?)

## 1.3 במידה ויש לו מתחרים

(מי הם ומה ההיתרון שלנו למולם)

אולי זה רעיון לכתוב כאן משהו על הonetcenter כמובן שזה לא כ"כ מתחרה והוא בכלל מכסה את אזור ארה"ב אבל בכל זאת יש כאן משהו שקיים ולמה אנחנו נותנים משהו טוב יותר

## 1.4 אולי סיכום קטן על הפרק

נתחיל מהבעיה הכללית.

קודם כל יש בעיה כללית בגיוס עובדים. כשאנשים רוצים לגייס עובד. הם לא תמיד יודעים מה התכונות שלו האם התכונות שלו מתאימות למה שהם שהם רוצים או לא. לכן. יש הרבה שיטות לברר מי עובד ומה התכונות שלו.

החברה שלנו מציעה פתרון מסוג חדש- הפתרון שלנו הוא שימוש ברשתות החברתיות כדי להכיר את התכונות של המועמד. הבסיס של הפתרון הזה הוא מחקרים שנעשו על פרופילים ברשתות חברתיות ועל התנהלות ברשתות החברתיות. המטרה הסופית שלנו היא לאפשר למעסיק לקבל תמונה כוללת על העובד בהתאם לפרופילים שלו ברשתות החברתיות.

לגבי מתחרים כמובן שיש הרבה שמנסים לאפיין אבל אנחנו חלוצים בתחום הזה של אפיון לפי לפי רשתות חברתיות.

לגבי פתרון חלקי או מלא כמובן זה פתרון חלקי אבל אנחנו מנסים שזה יהיה פתרון כמה שיותר קרוב לפתרון השלם.

אתר שנעזרנו בו. זה האתר של onetcenter. אתר זה מספק מידע על המון המון מקצועות התכונות הנדרשות לכל מקצוע ועוד המון מידע על המקצוע.

היתרון שלנו על פני הדבר הזה הוא - האתר הזה מספק רק מידע כללי לגבי המקצועות ופחות מאשר לנו לזהות מה התכונות הספציפיות של המועמד שלנו. יש גם אפשרות שלמעסיק יש רצון לעובדים עם תכונות שונות ממה שהוגדר ב onetcenter. אנחנו מאפשרים למשתמש להגדיר בעצמו את התכונות המועדפות עליו.

לגבי onetcenter אנחנו השתמשנו באמת בכמה סקריפטים שכתבנו כדי לאסוף את המידע משם.

המידע הזה יהיה זמין למשתמש גם באתר שלנו. במידה שהיה נראה לנו רלוונטי כמובן

# רקע תיאורטי

## 2.1 רשתות מחשבים

נסקור כאן בקצרה מצד עולם רשתות המחשבים את הפעולות הנדרשות ליצירת קשר בין ישויות בעולם האינטרנט.

תחילה נגדיר מושגים בהם נשתמש:

1. ישות – היא כל רכיב המחובר לרשת בעל כתובת IP
2. Session – שיחה בין ישויות ברשת האינטרנט, לצורך הקמת השיחה אנחנו חייבים לפתוח socket בכל צד, כלומר בצד הserver ובצד הclient
3. Socket – בהגדרתו מורכב מIP וPORT, למשל כמו צינור בין שני צדדים הרוצים לדבר ביניהם. אמון על העברת התכנים בין האפליקציות בצורה תקינה, כלומר שאם למשל עובדים עם skype ובמקביל הbrowser מבקש דף אינטרנט, חשוב לנו שהתוצאות יתקבלו למקום הנכון ולא יהיה בלבול בין התוכנות.
4. כתובת IP – היא כתובת בשכבה השלישית network layer, עבור כל ישות ברשת האינטרנט, המאפשרת לזהות ולתקשר איתה ברשת.
5. PORT – מוגדר בשכבה הרביעית transport layer, אמון על העברת המידע לאפליקציה הנכונה.

בעולם האינטרנט של ימנו רוב סוגי התקשורת עבור משתמשים היא מסוג client-server (על מודל זה נרחיב בהמשך), כלומר ישנו לקוח אחד או יותר המעוניינים לגשת לserver מסוים ולקחת ממנו נתונים או לחילופין לבקש ממנו לבצע כל מיני פעולות (כמו למשל לבצע חישובים המצריכים עיבוד טוב יותר מאשר הרכיבים במחשב הביתי).

כאשר אנחנו ניגשים להסביר את תהליך הקמת השיחה וקיום השיחה בפועל בין שני הישויות, ישנם הרבה פרטים שעליהם לא נרחיב כאן, אבל באופן יותר כללי אנחנו נדרשים לכך שהserver "יאזין" על PORT וכך הclient יוכל לגשת ולקבל את הנתונים הדרושים.



**4230**

**80**



## 2.2 מודל client – server

client-server modelהיא ארכיטקטורת תוכנה לחישוב מבוזר, אשר מגדירה את היחס בין תוכנות משתפות פעולה.

המודל מחלק את המשימות או עומס העבודה בין הserver שהוא למעשה ספק השירות של דרישות הלקוח, לבין הclient שהוא מבקש השירות, הלקוח.

כלומר ישנה הפרדה ביניהם באופן כזה שהclient לא צריך לדעת מה קורה עם הניהול מאוחר, למשל אם הוחלט לעבור לDatabase אחר, הserver חייב להיות מעודכן בכך אבל מצד הclient זה "שקוף".

מודל client-server הוא אחד מתצורות ההתקשרות הנפוצות ברשתות מחשבים שהפך לאחד מהרעיונות המרכזיים והרבה applications משתמשים בו.

הserver הוא מחשב בדר"כ חזק יותר מהמחשב הביתי המריץ תוכנה פסיבית, המאזינה על port ידוע (כמו למשל 80 עבור HTTP וכו') או port אחר שהוגדר מראש[[1]](#footnote-2) כלפי הclient ומחכה לקבל בקשות.

הclient לעומתו בדרך כלל מהווה את ממשק המשתמש הוא מופעל על ידי המשתמש ופונה -

לשרת כאשר הוא זקוק למידע או שירותים ממנו.

בדרך כלל, תוכנות השרת והלקוח רצות על גבי מחשבים שונים והתקשורת ביניהן מתבצעת על גבי

רשת מחשבים. עם זאת, תוכנות השרת והלקוח יכולות לפעול גם על גבי אותו מחשב. מכונת שרת

הוא מחשב המריץ תוכנת שרת אחת או יותר, אשר חולקת את המשאבים שלה עם הלקוחות.

הלקוח אינו חולק את המשאבים שלו, אלא מבקש תכנים ושירותים מהשרת.

לפיכך, הלקוח הוא זה שיוזם את ההתקשרות עם השרת, אשר ממתין לבקשות נכנסות.



צד לקוח

יישום הקיים בצד המשתמש (דפדפן, אפליקציות שונות, בין אם מחשב ובין אם טלפון חכם) והוא בדר"כ יוזם החיבור למול השרת לצורך ביצוע פעולות שהוא לא מסוגל לבצע, בין אם מדובר ביכולת חישוב או בגישה למאגרי נתונים (DB) שאין לו יכולת/הרשאה להגיע אליהם.

כל עמוד המוצג למשתמש מורכב למעשה מכמה חלקים:

1. קובץ css האחראי על עיצוב הדף
2. Script הכתוב בjavascript האחראי על הפונקציונאליות של הדף
3. קובץ HTML מגדיר את הדף והפקדים השונים ומכיל script וקישור לקובץ css

צד שרת

כאן מתבצעות הפעולות שמאחורי הבקשות של הלקוח, למשל אם ניקח את פרוטוקול HTTP (אחראי על העברת קבצי HTML) ובו ישנם פקודות GET ו-POST, כאשר הלקוח מקליד כתובת URL הוא למעשה יוצר פקודת GET וניגש לשרת שבמקרה שלנו מאזין על port 80 ומבקש ממנו דף אינטרנט מסוים, השרת מחפש את הדף המבוקש ומעביר אותו ללקוח.

ובמקרה בו הלקוח יבצע פקודת POST הוא מעלה נתונים לשרת והשרת כבר צריך לטפל בהם בין אם זה לבצע פעולות ולאחסן בבסיס נתונים וכו'.

https://www.ibm.com/support/knowledgecenter/en/SSAL2T\_9.1.0/com.ibm.cics.tx.doc/concepts/c\_clnt\_sevr\_model.html

https://en.wikipedia.org/wiki/Client%E2%80%93server\_model

http://webmaster.org.il/articles/how-does-server-side-works

<https://en.wikipedia.org/wiki/Client-side>

<https://skillcrush.com/2012/07/30/client-side-vs-server-side/>

## 2.3 ניתוח כלי פיתוח

### 2.3.1 סביבת עבודה

#### 2.3.1.1 pyCharm



סביבה המותאמת לשפת python (מאת Jet Brain) המאפשרת לכתוב תוכנית בצורה מסודרת ו/או ע"פ OOP.

קיים debugger המאפשר לנו להריץ את הקוד step by step ולצפות במשתנים.

מתריע על שגיאות syntax וקונבנציות 8PEP .

מקל על איתחול התוכנית לאחר קריסה או טעות.

<https://confluence.jetbrains.com/display/PYH/Getting+Started+with+PyCharm>

#### 2.3.1.2 notepad++



**Notepad++‎** הוא עורך טקסט ועורך קוד מקור ל.Windowsהתוכנה מציעה מגוון רחב של אפשרויות מעבר לפנקס רשימות המובנה בחלונות, כגון שימוש בלשוניות.

ממשק העשיר של Notepad++‎ מציע תכונות רבות לצרכים שונים, ביניהם:

* סימון וסידור תחביר בשפות תכנות שונות
* חיפוש והחלפה עם ביטויים רגולארים במספר מסמכים (תכונה חשובה ומשמעותית מאוד לטיפול בכמות גדולה של טקסט)
* ממשק נוח הכולל שימוש בלשוניות, גרור ושחרר, ומסך מפוצל
* תמיכה במגוון רחב של שפות וקידודים
* הגדלה והקטנה
* קיפול (collapse) ופתיחה של קטעי קוד
* השלמה אוטומטית
* תמיכה במאקרו ופלאג אין

<https://he.wikipedia.org/wiki/Notepad%2B%2B>

#### 2.3.1.3 Tableau

#### 2.3.1.4 Node.js command prompt

#### 2.3.1.5 Firebase Realtime Database

Firebase Realtime Database הוא מסד נתונים של NoSQL מסוג JSON שמבוסס על cloud. השירות מספק למפתחים API שמאפשר לחבר את התוכנה בקלות לDB. לfirebase יש ספריות שמאפשרות אינטגרציה לשפות פיתוח רבות.

<https://en.wikipedia.org/wiki/Firebase>

<http://meta-pa.blogspot.com/2017/08/firebase.html>

#### 2.3.1.6 Amazon Web Services



#### 2.3.1.7 GIT



Git, או בעברית גיט, היא מערכת ניהול גרסאות מבוזרת. על מנת להבין את המאפיינים והתכונות של גיט יש לחזור אחורה ולהגדיר מהי בכלל מערכת ניהול גרסאות מבוזרת. במילים פשוטות, ניהול גרסאות בא לידי ביטוי במעקב אחר וריאציות שונות של אותה יחידת מידע. השימוש בניהול גרסאות נעשה לא רק בהנדסה ובבניית אתרי אינטרנט אלא גם בפיתוח תוכנה, בכתיבת קוד מקור של יישומים, בדגמים אלקטרוניים ועוד. כלי תוכנה לניהול גרסאות חיוניים לכל פרויקט ומכאן גם הביקוש למערכת ניהול הגרסאות המבוזרת של גיט.

במודלי אחסון מבוזרים כל מפתח עובד עם מקום אחסון מקומי משלו. השינויים בסופו של דבר משולבים בין מקומות האחסון בשלב נפרד. כתוצאה ממצב הפעולה הזה יכולים המפתחים לעבור ללא חיבור רשת, כמו גם ליהנות מהיתרונות של יכולות ניהול גרסאות מלאות ללא צורך בקבלת הרשאות. לצורך השוואה, מודלי אחסון ריכוזיים מחייבים ביצוע של כל גרסאות הניהול על שרת משותף.

התכונות הבולטות של גיט

בתור מערכת ניהול גרסאות מבוזרת, גיט זוכה להצלחה בעיקר בגלל שהיא שומרת את תכני הקבצים עצמם. היא תוכננה במקור לשמש כמערכת קבצים משוכללת שעליה מערכת ניהול גרסאות, אך כיום היא מספקת גם מימוש אחיד של מערכות ניהול גרסאות. כמו כן, יש לציין שהמאגר של גיט נדרש לעבוד מול מאגרים אחרים. ההבדל הוא בין עבודה מול מאגרים שהם קבצים וחלק ממערכת קבצים מקומית, או קבצים מרוחקים על שרתים אחרים.

במידה והקבצים נמצאים על שרתים אחרים יכולים משתמשי מערכת ניהול הגרסאות המבוזרת גיט לבחור כיצד לגשת אליהם. אופציה אחת היא לעשות זאת באמצעות פרוטוקול "טיפש" שמיועד לקריאה בלבד (למשל HTTP), בעוד אופציה שנייה היא להשתמש בשרת גיט ייעודי. אופציה שלישית היא לבצע חיבור דרך SSH וזו גם האפשרות שנחשבת ליעילה ביותר.

<https://www.mrcoral.co.il/%D7%9E%D7%94-%D7%96%D7%94-git-%D7%92%D7%99%D7%98/>

#### tortoisegit –



תוכנה ייעודית לwindows שמספקת ממשק GUI נוח מאוד לכל הפעולות של GIT. היא מאפשרת גם את הצגת השינויים בצורה נוחה.

#### Github-



Github זה שרת מבוסס רשת לאחסון פרוייקטים שעובדים עם git.

עבור תוכנות של קוד פתוח, גיטהאב הוא חינמי. כל אחד יכול לפתוח שם חשבון ולהעלות אליו כמה מאגרים שירצה. הפרויקטים שמאוכסנים שם זמינים לכל מי שמעוניין, שיכול לשכפל אותם ולהשתמש בהם באופן אישי. לפעמים מגדירים את גיטהאב כ"רשת חברתית לתוכנות" שהפעילים בה יוצרים תוכנה או תורמים לתוכנה.

בנוסף, גיטהאב מכיל כלים נוספים לניהול הפרוייקט. אנחנו השתמשנו לא מעט בkanban board (נפרט בנפרד על מה מדובר), שאפשר לנו גם לעקוב אחרי המשימות שיש לנו, וגם לנהל את הטיפול בהן בצורה נוחה.

<https://internet-israel.com/%D7%A8%D7%A9%D7%AA-%D7%94%D7%90%D7%99%D7%A0%D7%98%D7%A8%D7%A0%D7%98/%D7%9C%D7%9E%D7%A4%D7%AA%D7%97%D7%99-%D7%95%D7%91%D7%95%D7%A0%D7%99-%D7%90%D7%AA%D7%A8%D7%99-%D7%90%D7%99%D7%A0%D7%98%D7%A8%D7%A0%D7%98/%D7%A4%D7%A8%D7%95%D7%A4%D7%99%D7%9C-github-%D7%94%D7%95%D7%90-%D7%94%D7%9B%D7%A8%D7%97%D7%99-%D7%9C%D7%9E%D7%A4%D7%AA%D7%97%D7%99%D7%9D-%D7%95%D7%9C%D7%9B%D7%90%D7%9C%D7%95-%D7%A9%D7%9E%D7%97%D7%A4/>

#### Kanban board

למילה היפנית קנבן יש מספר פרושים והפירוש הרלוונטי למקרה זה הוא כרטיס. בשורות הבאות ננסה להסביר את הרעיון ומקור השם.

הרעיון המרכזי הוא לקבל חומרי גלם או מוצר גמור בדיוק ברגע הנחוץ. לא לפני ולא אחרי.

נניח שיש לנו שרשרת יצור קצרה של 4 תחנות: לקוח, תחנת יצור, מחסן וספק כמו באיור להלן:



• בתחנת היצור מחזיקים מוצר אחד עבור הלקוח ובצד השני מגירה עם חלקים ליצור חלק נוסף בעת הצורך.

• הלקוח מושך מתחנת היצור מוצר גמור.

• העובד בתחנת היצור לוקח את חומר הגלם מהמגירה ומיצר מוצר חדש. במקביל הוא מעביר למחסן את המגירה הריקה עם כרטיס זיהוי שמזהה את התוכן שהיה במגירה.

• במחסן יש מגירה מוכנה עם חומר הגלם המיועד והמחסנאי מעביר את חומר הגלם ליצור.

• במקביל, העובד במחסן מעביר הזמנה לפי כרטיס הזיהוי לספק שמספק מגירה חדשה למחסן.

המספור של המרובעים באיור אינו מקרי וכך גם החיצים עם שני הראשים. הכוונה היא להדגיש שעובדים בשיטה של משיכה ולא דחיפה.

הסוד במערכת קנבן טובה הוא לחשב כמה כרטיסים נדרשים לכל מוצר. רוב המפעלים משתמשים בשיטת הלוח הצבעוני (Heijunka Box). שיטה מורכבת מלוח הנוצר במיוחד על מנת להחזיק את כרטיסי הקנבן.

בשנים האחרונות קנבן אומצה גם בעולם פיתוח התוכנה, כשיטה המשלבת עקרונות מעולם ה-Agile וה-Lean, כאשר מרכז השיטה הוא אותו רעיון של הגבלת הקנבנים שבמערכת, ועל ידי כך הורדת מלאי הנושאים שמפותחים ונבדקים במקביל ב"קו הפיתוח". השיטה נקראת Kanban System for Software Engineering

<https://he.wikipedia.org/wiki/%D7%A7%D7%A0%D7%91%D7%9F>

<https://business-excellence.co.il/blog/513-kanban-system>

<https://he.wikipedia.org/wiki/%D7%A7%D7%A0%D7%91%D7%9F>

<https://en.wikipedia.org/wiki/Kanban_(development)>

#### Chrome DevTools

Chrome DevTools הוא אוסף של כלים למפתחי אינטרנט, שמובנים בדפדפן Google Chrome.

תכונות מרכזיות שהכלים האלה מאפשרים:

- לצפות בקוד HTML/CSS עבור חלק נבחר מהעמוד

- לעשות שינויים בקוד HTML/CSS בעמוד

- לדבג קוד js בconsole, וכן לצפות בשגיאות שקורות במהלך הריצה של קוד js.

- לצפות בקבצים שמהם מורכב העמוד שבו אנחנו צופים, וכן לעקוב אחרי התעבורה ברשת שנדרשה עבור קבלתם.

<https://developers.google.com/web/tools/chrome-devtools/>

<http://www.bekaloot.co.il/%D7%9E%D7%93%D7%A8%D7%99%D7%9A-2321-%D7%90%D7%99%D7%9A%20%D7%9C%D7%93%D7%91%D7%92%20%D7%90%D7%AA%D7%A8%20%D7%90%D7%99%D7%A0%D7%98%D7%A8%D7%A0%D7%98%20%D7%91%D7%A2%D7%96%D7%A8%D7%AA%20%D7%92%D7%95%D7%92%D7%9C%20%D7%9B%D7%A8%D7%95%D7%9D.aspx>

#### 2.3.1.8 JSON

#### 2.3.1.9 HTTP

### 2.3.2 שפות פיתוח

#### Javascript 2.3.2.1

היא שפת תוכנה (ללא כל קשר לשפת java) ליצירת אתרים דינאמיים, בהם השינויים באתר ובדף האינטרנט לא מצריכים טעינה של הדף מחדש אלא מורצים בצד הלקוח.

בניגוד לשפות כמו Java שמתורגמות לשפת מכונה (compiled), שפת JavaScript מפורשת על ידי הדפדפן (או כל כלי אחר) בזמן אמת. זה הופך את השפה ליותר איטית בהשוואה לשפות מתורגמות אך זהו עניין זניח כשמדובר באתרי אינטרנט שרובם ככולם לא דורשים עיבודים מורכבים.

בתחילת דרכה, עיקר השימוש של השפה היה הרצת קוד בתוך הדפדפן בצד הלקוח. אולם מספר שנים לאחר מכן השפה פרצה את תחום הדפדפנים ועברה להיות שפה עצמאית שניתן ליצור איתה הכל, ואפילו להשתמש בה לתכנות צד שרת. הדבר נעשה אפשרי אודות לפיתוח סביבת ההרצה הנקראת [Node.js](https://nodejs.org/). בעזרת Node ניתן להריץ JavaScript ללא צורך בדפדפן. Node משתמש במנוע להרצת JavaScript שפותח על ידי Google ונקרא V8. על ידי שימוש בNode ניתן לפתח אפליקציות מבוססות JavaScript למכשירים ניידים ונייחים ולמגוון מערכות הפעלה.

<https://he.wikipedia.org/wiki/%D7%90%D7%AA%D7%A8_%D7%90%D7%99%D7%A0%D7%98%D7%A8%D7%A0%D7%98>

<https://skillcrush.com/2012/04/05/javascript/>

#### HTML 2.3.2.2

HTML  (Hyper Text Markup Language) הוא פורמט שמשמש לבניית אתרים.

דפי HTML הם קבצים עם טקסט פשוט, אשר מסביבו הוראות הנמסרות לדפדפן ומורות לו כיצד להציג את הדף. כמו למשל 'הדגש את הטקסט', 'הטקסט הזה הוא כותרת ראשית', 'הטקסט הזה הוא קישור לדף אחר'.

הוראות אלו הן מה שהופך קובץ טקסט רגיל לHTML והן נקראות תגים (tag). מלבד התגים, דבר נוסף המאפיין דפים כאלו הוא סיומת הקובץ שלהם, שהיא: "html." או "htm.".

תגים הם הבסיס של HTMLבעזרתם יכול כותב האתר לשנות את מה שמוצג על המסך. תגים הם בעצם פקודות שהדפדפן מקבל ומנסה לבצע אותן (במידה והוא מכיר את התג) - ישנם תגים לשינוי צבע הטקסט על המסך, לשינוי המיקום, ליצירת טבלאות, לשימוש בתמונות, ואפילו תגים המכילים רשימות מילות מפתח בשביל מנועי חיפוש ועוד ועוד.

ניתן לרשום את כל התגים בשורה אחת, וניתן לרשום כל תג בשורה נפרדת. החשיבות היחידה לכך היא שבכתיבה בשורות נפרדות קל יותר ונוח יותר לקרוא ולהבין את המסמך.

שלבי השימוש בתג:

* מתחילים בתג פותח <tagname> בתג הפותח ניתן למצוא את שם התג, מוקף בסוגריים משולשים <> במקרה הזה, שם התג הוא: “tagname”.
* כותבים את תוכן התג תוכן. התוכן של התג הוא מה שמושפע ממנו.
* מסיימים בתג סוגר </tagname> תג סוגר הוא תג שמוסיפים בו לוכסן רגיל לפני שם התג. חובה להשתמש בו, ואם לא משתמשים בו התג לא יפעל.

שלושת השלבים הללו, התג הפותח, התג הסוגר והתוכן, נקראים ביחד אלמנט. שם האלמנט הוא שם התג. במקרה זה, שם האלמנט הוא: “tagname”.

ישנם כמה סוגים של תגיות:

* **תגיות מבניות**: תגיות המשמשות בעיקר להוספת טקסט לעמוד.

לדוגמה: <p/> The Best Project<p> התגית הנ"ל תוסיף את הטקסט

" The Best Project " לעמוד, ותגדיר אותו כפסקה (P=Paragraph).

* **תגיות תצוגה**: מגדירות מאפייני תצוגה שונים עבור אלמנטים בעמוד, דוגמת טקסט. המאפיינים השונים יכולים להיות סוג גופן, הדגשות, צבע גופן ועוד.

לדוגמה: <b/>Build Cool Web!!<b>. תגית זאת תוסיף את הטקסט

" Build Cool Web!!" באופן מודגש (b=Bold) לעמוד. למרות זאת, כיום מקובל לבצע הגדרות אלה לא באמצעות תגיות html, אלא באמצעות תבניות עיצוב ([CSS](http://www.xn--7dbbqer5d.co.il/dreamweaver/css.html)) עליה נדבר בהמשך.

* **תגיות קישורים**: תגיות המשמשות ליצירת קישורים מעמוד האינטרנט לעמוד אחר, או למיקום אחר באותו העמוד. לדוגמה: <a href=http://google.co.il>Google</a>. תגית זאת תיצור קישור שייראה כך בעמוד האינטרנט: Google
* **תגיות הוספה**: תגיות המשמשות להוספת אלמנטים שונים לעמוד, דוגמת תמונות, קבצי מדיה ועוד. לדוגמה: </"img src="FinalPro.jpeg> . תגית מסוג זה תציג בעמוד האינטרנט את קובץ התמונה ששמו FinalPro.jpeg
* **תגיות טפסים**: תגיות ליצירת והוספת טפסים לעמוד האינטרנט.
* **תגיות תוחמות**: הן תגיות שיוצרות מן מתחם בעל יעוד מסוים אשר מכיל בתוכו קבוצת תגיות נוספות. דוגמה לתגיות אלה הן תגית ה <head> וה <body> וכן תגיות ה <div> אשר יוצרות תת מתחם בתוך עמוד האתר בו ניתן לשים אובייקטים שונים ותוכן.

<https://en.wikipedia.org/wiki/HTML>

<https://www.w3schools.com/html/html_intro.asp>

#### CSS 2.3.2.3

דפי אינטרנט נכתבים בשפת [HTML](https://he.wikibooks.org/wiki/HTML) כאשר כל דף HTML מכיל אלמנטים בתגי HTML , ניתן לשייך לכל אלמנט בנפרד הגדרות עיצוב ע"י מאפייני האלמנט אך זה יהיה עיצוב שגוי של האתר, כי נניח שהאתר כולל 20 מסמכי HTML ונניח שכל מסמך מכיל 20 אלמנטים. ולכל אלמנט אנו נשייך הגדרות עיצוב. אז יהיו לנו 400 הגדרות עיצוב.

ובד"כ, אתר אינטרנט, שומר על אחידות בעיצוב שלו בין הדפים השונים, אז כפועל יוצא, נקבל הגדרות עיצוב שתחזורנה על עצמן, כלומר שכפול של קוד(=עיצוב שגוי). CSS באה לפתור את הבעיה הזו: נגדיר קובץ CSS (בד"כ אחד או שניים), שיכיל את כל הגדרות העיצוב של האתר, למשל, שכל הפסקאות <p> יהיו עם font מסוג DAVID וכך נמנע משכפול של קוד.

וכן, אם בעתיד נדרש לבצע שינוי בעיצוב של האתר, נצטרך לבצע זאת רק בקובץ הCSS ולא בכל דפי ה HTMLשהגדרנו.

#### Python2.3.2.4

היא שפה עילית המפותחת בקוד פתוח , [שפת תכנות דינמית](https://he.wikipedia.org/wiki/%D7%A9%D7%A4%D7%AA_%D7%AA%D7%9B%D7%A0%D7%95%D7%AA_%D7%93%D7%99%D7%A0%D7%9E%D7%99%D7%AA) ומונחת עצמים עם ניהול זיכרון אוטומטי הניתנת להרחבה ע"י שימוש במודולים, תוכננה תוך שימת דגש על קריאוּת הקוד, וכוללת מבנים המיועדים לאפשר ביטוי של תוכניות מורכבות בדרך קצרה וברורה. אחד המאפיינים הבולטים בתחביר השפה הוא השימוש ב[הזחה](https://he.wikipedia.org/wiki/%D7%94%D7%96%D7%97%D7%94) להגדרת בלוקים של קוד (ללא שימוש בסוגריים או ב[מילים שמורות](https://he.wikipedia.org/wiki/%D7%9E%D7%99%D7%9C%D7%94_%D7%A9%D7%9E%D7%95%D7%A8%D7%94) לצורך כך, כמו ברוב השפות הנפוצות).

פותחה בשנת 1991 ע"י גואידו ואן רוסום, במקור בתור פיתוח של [שפת התכנות ABC](https://he.wikipedia.org/wiki/ABC_(%D7%A9%D7%A4%D7%AA_%D7%AA%D7%9B%D7%A0%D7%95%D7%AA)) אשר יוכל [לטפל בחריגות](https://he.wikipedia.org/wiki/%D7%98%D7%99%D7%A4%D7%95%D7%9C_%D7%91%D7%97%D7%A8%D7%99%D7%92%D7%95%D7%AA_(%D7%AA%D7%9B%D7%A0%D7%95%D7%AA)) עבור[Amoeba OS](https://he.wikipedia.org/w/index.php?title=Amoeba_OS&action=edit&redlink=1), פותחה מתוך רצון להגיע לשפה פשוטה ומובנת, נוחה לקריאה וקלה לתחזוקה.

השפה מורצת ע"י interpreter המפרשן כל שורה בנפרד ומריץ אותה (לעומת שפות כמו C שעוברות קומפילציה בה הכל מפורש לקוד מכונה בפעם אחת ומורץ)

אפשר לעבוד אינטראקטיבית במעטפת פקודה כשפה סקריפטית. לאחר הקלדת כל פקודה, מקבלים ישירות את תוצאת המפרש.

תכונה זאת מאפשרת למידה קלה ותכנות מהיר עם Debug תוך כדי כתיבה. Python

היא שפה בעלת ניידות גבוהה, כך שניתן להריצה על מערכות רבות Apple OS/X, Linux Windows, Unix הקוד ב- Python הרבה פחות "מלוכלך" מסימני תחביר כגון הצהרות משתנים.

קוד ב- Python לרוב קצר פי 3-5 מקוד מקביל ב- Java ופי 5-10 מאשר קוד מקביל ב- C++ .

בקיצור, Python טובה לכל צורך של כתיבת קוד מהירה ויעילה.

Python 2 vs Python 3

Python 3.0 שוחרר בשנת 2008 . הגרסה האחרונה של Python 2 שהיא גירסת 2.7 יצא לאור בשנת2010 לאחר מכן Python 2 לא הוציאה שום גרסאות חדשות עיקריות, בניגוד ל- Python 3 שלא הפסיק להתפתח.

פיתוח זה כולל גירסת 3.3 בשנת 2012 , 3.4 ב- 2014 וגירסת 3.5 בשנת 2015 . משמעות

הדבר היא שכל השיפורים האחרונים של הספרייה הרגילה למשל (standard library) אך ורק זמינים כברירת מחדל ב- Python 3 כמה היבטים של ליבה של השפה כמו למשל exec הותאמו להקלת לימוד השפה במיוחד עבור מפתחים חדשים, ולהיות בקנה אחד עם שאר השפה.

גירסה 2.x של Python היא מורשת בעוד שהגירסה 3.x היא ההווה והעתיד של

השפה. לכן אנחנו השתמשנו בגירסת Python 3 במהלך הפרוייקט.

#### Database

אמצעי המשמש לאחסון מסודר של [נתונים](https://he.wikipedia.org/wiki/%D7%A0%D7%AA%D7%95%D7%A0%D7%99%D7%9D) ב[מחשב](https://he.wikipedia.org/wiki/%D7%9E%D7%97%D7%A9%D7%91), לשם אחזורם ועיבודם. בסיס נתונים מאוחסן ב[אמצעי אחסון נתונים](https://he.wikipedia.org/wiki/%D7%90%D7%9E%D7%A6%D7%A2%D7%99_%D7%90%D7%97%D7%A1%D7%95%D7%9F_%D7%A0%D7%AA%D7%95%D7%A0%D7%99%D7%9D), בדרך כלל על גבי [דיסק קשיח](https://he.wikipedia.org/wiki/%D7%93%D7%99%D7%A1%D7%A7_%D7%A7%D7%A9%D7%99%D7%97), המאפשר גישה ישירה לנתונים.

בסיס הנתונים בנוי לפי מודל לאחסון הנתונים: רשתי, היררכי, טבלאי, מונחה עצמים וNoSQL המבטאים סוגי קשרים שונים בין היישויות השונות.

המודל הנפוץ ביותר היום (והוא המודל בו השתמשנו), הוא המודל הטבלאי שמכונה גם רלציוני.

במודל זה בסיס הנתונים בנוי מטבלאות, כאשר כל טבלה מכילה מידע על ישות מסוימת, ושורה מבטאת רשומה, כאשר כל רשומה מתייחסת למקרה ספציפי (למשל תלמיד מתוך מאגר של מוסד אקדמי).

השדה חייב להיות מסוג מסוים, כלומר כאשר נגדיר עמודה בטבלה המייצגת את גילו של התלמיד היא תיהיה מסוג int[[2]](#footnote-3) , במידה ונרצה להגדיר עמודה עם שמו של התלמיד היא תיהיה מסוג char וכו'.

לכל רשומה בטבלה יש [מפתח ראשי](https://he.wikipedia.org/wiki/%D7%9E%D7%A4%D7%AA%D7%97_%D7%A8%D7%90%D7%A9%D7%99) שמזהה באופן ייחודי את הרשומה (כמו למשל מס' תלמיד או תעודת זהות).

הקשר בין הרשומות בטבלאות שונות נעשה באמצעות שדה [מפתח זר](https://he.wikipedia.org/wiki/%D7%9E%D7%A4%D7%AA%D7%97_%D7%96%D7%A8) כאשר מספר הפעמים בו יכול להיות משויך המפתח לרשומה ספציפית בטבלה האחרת מגדיר את ריבוי הקשר (יחיד לרבים - רבים לרבים וכו').

בסיס נתונים מכיל בדרך כלל רשומות רבות, המקושרות באמצעות מפתחות. לתרשים שמבטא את כל הקשרים בין הרשומות השונות קוראים דיאגרמה. היחסים בין רשומות יכולים להיות מכמה סוגים. הנפוצים ביותר הם יחס של אחד לרבים, שבו לכל רשומה יכולות להיות מקושרות מספר רשומות אחרות, ויחס של אחד לאחד, שבו לכל רשומה יכולה להיות רק רשומה אחת שמקושרת אליה, למשל תלמיד מקושר לכמה מרצים.

שליפת מידע ופעולות עדכון בבסיס נתונים טבלאי נעשות באמצעות שפת [SQL](https://he.wikipedia.org/wiki/SQL), המהווה ממשק המאפשר גישה לנתונים מבלי להתייחס לאופן שמירתם בבסיס הנתונים.



<https://he.wikipedia.org/wiki/%D7%91%D7%A1%D7%99%D7%A1_%D7%A0%D7%AA%D7%95%D7%A0%D7%99%D7%9D>

<https://en.wikipedia.org/wiki/Database>

<https://searchsqlserver.techtarget.com/definition/database>

#### postgreSQL2.3.2.5



PostgreSQL היא מערכת לניהול בסיס נתונים רלציוני, מדובר במערכת open source חינמית ואינה נשלטת על ידי אף חברה מסוימת אחת, אלא נתמכת על ידי אוניברסיטת קליפורניה והקהילה הגלובלית של מתכנתים וחברות שיפתחו אותה(מעל 30 שנה של פיתוח)PostgreSQL מוטמעת כתת-מערכת נרחבת של ANSI SQL ויכולה לרוץ על רוב מערכות ההפעלה. כמו כן, היא מתממשקת עם שפות תכנות רבות .

קיימות לפוסטגרס הרחבות רבות, והרבה תכונות שעוזרות למפתחים לכתוב תוכנות בצורה פשוטה יותר, בניהול שלימות המידע ובסביבה קלה לתפעול.

בשל כך פוסטגרס , בסיס הנתונים במודל הרלציוני הפך להיות בסיס הנתונים הרלציוני הפופולארי בקרב הרבה אנשים וארגונים

<https://www.postgresql.org/about/>

## 2.4 מודל הפיתוח

להציג כמה מודלים ובינהם את סקרם ולכתוב מדוע בחרנו במודל הזה

ברחנו בscrum

# תכנון

## 3.1 דרישות המערכת

### 3.1.1 דרישות כלליות

### 3.1.2 דרישות side client

### 3.1.3 דרישות side server

## 3.2 אפיון המערכת

## 3.3 מבנה המערכת (אופציונאלי – לא בטוח שזה יופיע בדווקא כאן)

המערכת מורכבת משרת node js,ו-DB של SQL. כדי לגשת למערכת יש להשתמש בדפדפן רגיל. המשתמש ניגש באמצעות הדפדפן אל השרת. השרת מחזיר לו דף אינטרנט שכולל את המקצועות המתאימים. לאחר בחירת מקצוע, השרת ניגש לDB, מביא את כל הנתונים הקיימים כבר על המקצוע, ושולח אותם לדפדפן. המשתמש יכול להתחיל לערוך את פרטי המקצוע בשני דפים מרכזיים – הדף הראשון מכיל metadata לגבי המקצוע - שם התפקיד, תיאור התפקיד, והעדפות בסיסיות של החברה (למשל – אנשים מעל גיל 30). הדף הראשון מכיל את רשימת כל התכונות, כאשר עבור כל תכונה מופיע סרגל עם חמישה חלקים. כל חלק מייצג רמה אחרת של התכונה. למשל, אם מדובר בתכונה "פתיחות", אז קצה אחד של הסרגל מייצג פתיחות נמוכה, וקצה שני מייצג פתיחות גבוהה. עבור כל אחת (!) מהרמות, המגייס יכול להגדיר עד כמה רמה כזו רצויה לו (ניתן לתת ציונים בטווח 1-5). לאחר שהמתמש מסיים להזין את הנתונים, הוא לוחץ על כפתור, והנתונים נשלחים מהדפדפן לשרת. השרת מעדכן את הנתונים בDB.

# תכנון ביצוע

## 4.1 תחילת העבודה

בהתחלה הקמנו שרת בסיסי, ובנינו דף שמתחבר אליו.

המטרה שלנו הייתה לייצר את המערכת בצורה מינימלית כדי שנוכל להתחיל לראות לאיפה להתקדם.

הקמנו שרת פשוט ובנינו דף HTML שאותו השרת יחזיר.

הקמנו בסיס נתונים מקומי על המחשב בתור התחלה ויצרנו חיבור בין השרת לבין הבסיס נתונים.

בדף שבנינו עשינו חמישה כפתורי רדיו שייצרו את האפשרות לבחור חמש ציונים שונים לכל תכונה, או לכל חלק של תכונה.

הדף שולח את הבחירה לשרת והשרת שומר את זה בבסיס הנתונים.

עכשיו התחלנו להתעסק קצת בעיצוב של הדף. הוספנו בדף חמישה מרובעים, שכל מרובע מייצג חלק אחר של התכונה.

בלחיצה על מרובע מופיעים חמישה פרצופים, כשניתן לבחור בפרצוף המתאים שמייצג את היחס שלנו אל התכונה הזאת. למשל, ניתן לבחור שפתיחות גבוהה היא רצויה מאוד ופתיחות נמוכה היא לא רצויה.

בשלב הבא הוספנו טבלה שתשמור את הנתונים שנבחרו בדף.

הדף ישלח את הנתונים מהטבלה אל השרת

כשסיימנו את זה התחלנו להכניס את הנתונים מהשרת אל בסיס הנתונים.

השלב הבא היה. שהשרת מבקש מידע מבסיס הנתונים לפני שהוא מחזיר את הדף. בהחזרת הדף השרת מצרף את הנתונים שקיבל מבסיס הנתונים.

אחר כך הקמנו בסיס נתונים חדש באינטרנט בAWS וחיברנו אותו לשרת שלנו.

## 4.2 מודל הפיתוח של המערכת (כאן כדאי להציג מודלים – כדאי לעיין גם בפרויקט של חירום בפיקוד העורף )

## UI 4.3

בהתחלה התבקשנו לבצע את המשימה כאשר הfront end יהיה מיוצג ע"י tableau. התחלנו להכין את המערכת בעזרת קוד מינימלי מגיטהאב שהכיל שרתnode js שמדבר עם הדף שמוצג בtableau מצד אחד, ועם הDB מצד שני. בהמשך, ישבנו עם מנהל החברה, והצגנו לו את המשמעויות השונות – הערכנו שהשימוש בtableau כfront end עלול לדרוש מאיתנו לתחזק מספר גדול של דפים שישובצו במקומות המתאימים בדף שיוצג. בנוסף, זה דרש גם קניית רשיון עבור שימוש בtableau (כיוון שמדובר בחברת סטרטאפ לפני השלב של גיוס הכספים, היה מדובר בשיקול משמעותי). לבסוף התקבלה החלטה לממש את הfront end בעזרת דף html. בהתחלה בנינו דף HTML שעיצבנו לגמרי בעצמנו (כמובן בהתייעצות עם החברה). לאחר שסיימנו, מנהל החברה פנה אלינו וביקש לשנות קצת את העיצוב, בהתאם לחשיבה מחדש שנעשתה בחברה. הוא שלח לנו תמונות של עיצוב סופי כפי שהוא מעוניין. אנחנו פשוט התאמנו את הדף שלנו, כדי שיקבל את העיצוב המדוייק שהתבקשנו.

## 4.4 הקמת שרת Node.js

כפי שתיארנו, התכנון המקורי היה לממש back end עבור tableau, ולכן השתמשנו בבסיס של קוד שהיה קיים כבר עם node js. גם לאחר ההחלטה שלא להשתמש בtableau, החלטנו להמשיך עם node js. כך גם הרווחנו את ניצול הזמן שכבר השקענו בשרת, וגם מבחינתנו היה לזה יתרון, שנוכל להשתמש יותר בקוד javascript, שלא הכרנו קודם לכן, ולמדנו רק במהלך הפרוייקט.

## 4.5 הקמת instance in AWS

החברה משתמשת בשרת SQL בAWS. כדי להכין את המערכת בצורה מתאימה ככל האפשר, בנינו גם כן שרת SQL על AWS, ובעזרתו ניהלנו את המערכת שלנו.

## 4.6 ארגון בסיס הנתונים (אולי צריך לשלבו עם 4.5)

בהתחלה השתמשנו בטבלה אחת שתכיל את המידע על כל התכונות, וקצת metadata על התפקיד, כך שכל מקצוע תופס שורה אחת בטבלה. לאחר שסיימנו, התברר לנו שבחברה החליטו לשנות את המבנה של הDB, ויצרו טבלאות שונות לכל דבר – טבלה של תכונות, שבה כל תכונה נמצאת בשורה נפרדת, ושתי טבלאות עם metadata.נאלצנו לשנות קצת את הקוד, אבל בגלל שחילקנו את הקוד בצורה יחסית מודולרית, זה לא דרש מאיתנו יותר מדי מאמץ, ברוך ה'.

## Data Mining4.7 (אולי גם את זה נוסיף פשוט ל4.5)

מצאנו אתר שיש בו המון המון מידע על מקצועות. רצינו להציג את המידע הזה למשתמש כדי לעזור לו להגדיר את המקצועות שלו.

לכן אנחנו צריכים לגשת לאתר. להוריד ממנו את המידע. ולצרף אותו אלינו.

ראשית, ניסינו להשתמש בAPI הרשמי של האתר, אבל ראינו שצריך לחכות לקבלת הרשאה מהאתר בשביל זה. לכן החלטנו לגשת ישירות לקבצים שמעניינים אותנו. כתבנו סקריפט שניגש לכל הקבצים הרלוונטיים (היתה שם אפשרות להוריד קובץ עבור כל טבלה. אנחנו השתמשנו בזה, והורדנו את הקבצים בפורמט טקסט), מפרש את התוכן שלהם, ומחלץ מתוכם את שמות התכונות. בעזרת הפלט של הסקריפט הזה, יצרנו קובץ עם שמות כל התכונות הרלוונטיות שיש באתר. בהתאם, יצרנו טבלה בDB שתכיל את שם המקצוע, וציון עבור כל תכונה. אחר כך יצרנו קוד שניגש להורדת הקבצים, קורא אותם, וממיר את כל המידע לפורמט JSON, כאשר ה key הוא הID של המקצוע, ובvalue יש רשימה של תכונות, עם ציון עבור כל אחת (הקבצים האלה לא הכילו את שמות המקצועות, אלא רק את הID שלהם). אחר כך השתמשנו בקוד הזה כדי להוריד קובץ שמכיל את ההתאמה בין הID של המקצוע לשם המקצוע. בשלב הבא יצרנו סקריפט נוסף, שקורא את המידע מהקובץ שמכיל את הID של המקצועות ואת הציונים, ובמקביל קורא גם את הקובץ שמתאים בין ID לשם המקצוע. עבור כל מקצוע, הוא לוקח את הציונים עבור כל התכונות, מוסיף להם את שם המקצוע, ושומר אותם בטבלה שהכנו בDB.

כיוון ששמרנו את הנתונים בפורמט JSON, רצינו לשמור את כל הנתונים בfirebase. הכנו את הקוד המתאים, וגם העלינו את הנתונים לDB של firebase, אבל מנכ"ל החברה ביקש שנעביר הכל לpostgres, יחד עם הDB הראשי של החברה, כיוון שזה יחסוך להם בניהול המידע.

## 4.8 תהליכים

הגדרת נתונים לתפקיד

המשתמש ניגש לדף הראשי –



ראשית, עליו לבחור קטגוריה (מסומן בחץ כחול). לאחר בחירת הקטגוריה, נפתחות לו שתי אפשרויות – הוספת תפקיד חדש, או בחירת תפקיד קיים.



נדגים קודם כל את האפשרות של הוספת תפקיד חדש:



לאחר שהמשתמש לחץ על הכפתור שמסומן בחץ כחול, נפתחת לו תיבת טקסט (מסומנת בחץ ירוק) להזין לתוכה את השם של התפקיד החדש.



לאחר שהוא מסיים להזין את השם החדש, הוא לוחץ על הכפתור ADD, שמימין לתיבה. לאחר כמה שניות הוא מקבל מסר שהשם נוסף בהצלחה –



המסר נעלם לאחר כמה שניות.

כבר כעת אפשר לבחור את התפקיד הזה ברשימה הנפתחת של שמות התפקידים –



עכשיו יש למשתמש אפשרות להכניס נתונים בנוגע למקצוע (ניתן לשים לב בצד שמאל לID שיצרנו באופן אוטומטי):



מכאן והלאה, בכל שלב של מילוי הטופס, המשתמש יכול ללחוץ על submit, והנתונים שהזין בינתיים יישמרו. לדוגמא:



בשלב הזה המשתמש לוחץ על submit, והנתונים נשלחים לשרת ומשם לDB. הדפדפן מציג שוב את הדף ההתחלתי. לאחר בחירת המקצוע שאליו הזנו את הנתונים, הדפדפן שולח בקשה לשרת, שמביא לו מהDB את הנתונים לגבי אותו מקצוע. הדפדפן ממלא בהתאם את השדות המתאימים (ושוב, הדף נראה כמו בתמונה למעלה...). בצורה כזו המשתמש יכול לעצור את העבודה בכל שלב, ולחזור אליה מאוחר יותר.

לאחר שהמשתמש סיים להגדיר metadata, הוא ירצה לגשת להגדרת התכונות (ניתן לגשת לשם גם לפני כן, אין מגבלה בנושא). לצורך זה הוא ילחץ על הכפתור המסומן כאן בחץ:



לאחר הלחיצה הוא מועבר לדף הבא:



בצד שמאל למעלה ניתן לראות כפתור לחזרה לדף הקודם.

הדף עצמו מספק אפשרות לבחור רמות רצויות עבור תכונות מסויימות. רק נציין שכל התכונות מוצגות ברצף אחת מתחת לשניה:



לאחר שהמשתמש לוחץ על אחד מהריבועים, הריבוע הזה מסומן במסגרת כחולה, ומופיע מסר שמסביר לו מה הוא מגדיר כעת (במקרה שלנו – עד כמה הוא מעוניין במועמדים עם רמת פתיחות ממוצעת):



לאחר לחיצה על אייקון מסויים, המרובע המתאים ייצבע בצבע של האייקון, כך זה נראה לאחר שהמשתמש מילא את כל הסרגל:



לאחר שהמשתמש מסיים להגדיר את כל התכונות, הוא לוחץ על submit, והכל נשמר בDB. כמובן, בפעם הבאה שיבחר במקצוע הזה, הדפדפן יציג לו שוב את הצבעים שכבר בחר (על פי מה שנשמר בDB)

## 4.9 פונקציונאליות

## 4.10 בדיקות

# 5. דיון

## 5.1 יתרונות האפליקציה

## 5.2 הרחבות אפשריות למערכת

## 5.3 רווחים אישיים מהפרויקט

# 6. סיכום

# 7. ביביליוגרפיה

-

1. כיוון שכדי שהclient יוכל ליצור קשר למול הserver שלנו הוא נדרש לדעת לאיזה port לגשת [↑](#footnote-ref-2)
2. כמובן בהנחה שלא מתייחסים לחישוב החודשים אלא למס' עגול המבטא את גילו של התלמיד [↑](#footnote-ref-3)