



החוג להנדסת מערכות תקשורת

פרוייקט גמר

**מערכת חכמה להתאמה מקסימלית בין מועמד למשרה**

**מגישים:**

**אסף גמליאל [ת.ז]**

**דניאל דנן [ת.ז]**

מנחה אקדמי:

מר יאיר גולדשטיין

אדר ב' תשע"ט

תודות

תחילה נודה לקב"ה על שחנן אותנו ביכולות וזיכה אותנו ללכת בדרכה של תורה, ומקווים אנו להועיל לעולם ובפרט לעם ישראל בכלים שרכשנו לאורך שנות הלימוד במכון לב.

תודות מיוחדות למשפחותינו ולנשותינו שתמכו וסייעו לנו לסיים את שנות הלימודים בהצלחה

תודה למכון לב ולצוות הרבנים והמרצים ובפרט למנחה שלנו (יאיר) שהכווין אותנו בצמתים בהם נתקלנו במהלך הפרויקט

## תקציר

כיום מלאכת גיוס העובדים היא לא דבר פשוט, בחברות רבות ישנם תהליכים ומתודות שונות איך לנהל את משאבי האנוש ולהביא לגיוסם של עובדים מוצלחים שיתנו תועלת לחברה בה הם עובדים.

ישנם כלים/פרמטרים שונים בהם המועמדים נבחנים, כמובן כל אחד לפי תכונותיו האישיות ויכולותיו הביצועיות ובהתאם למשרה המוצעת.

למשל בחברות ממשלתיות כמו רפאל המועמד עובר הליך סינון ראשוני בו הוא נמדד ע"פ ציוניו וקו"ח אשר הוא הגיש, לאחר מכן מתקיים ראיון טלפוני ובמידה והולחט על המשך, הוא יגיע לראיון אישי ולאחריו ליום מבחנים במכון בו הוא יעבור מבחנים פסיכוטכניים, שיחה למול פסיכולוג ומשימות קבוצתיות.

הדוגמה שהבאנו באה לסבר את האוזן על התהליך שחברה צריכה לבצע למועמד עד להחלטה הסופית בה היא מעוניינת להעסיק את המועמד.

קל לראות שמדובר בהליך ארוך מבחינת זמן, דבר שיכול לגרום למועמד טוב לחפש במקביל משרה אחרת ולחכות לראשון מבינהם (ולמעשה בכך החברה אולי מפסידה מועמד טוב, זמן ומאשבים שהושקעו בו) וגם מדובר בהשקעה כלכלית כלשהי שיכולה לרדת לטמיון.

חברת Cyber Traits ראתה לנכון להכניס רוח חדשה ולהפוך את התהליך לקל, זול, חכם ומדויק יותר עבור המגייסים.

הפתרון מבוסס על שנות מחקר רבות של מנכ"ל החברה פרופ' יונתן (בוגר הארווארד), אשר אמון גם על השאלונים והמבדקים הנעשים בצבא.

על מחקרים אלו אנחנו מתבססים בפיתוח המערכת אשר נבנה בתוכה מאגר נתונים גדול עבור תכונות אישיות שונות ביחס לתפקידים שונים בתעשייה, בו יש יכולת למגייסים לשנות את "המינונים" השונים של תכונותיו של המועמד ביחס לתפקיד או לחילופין ליצור תפקידים חדשים אשר לא קיימים במאגר ולהגדיר את התכונות הרצויות

בשלב זה הקלט עבור המערכת הוא שאלונים מיוחדים אותם כתב מנכ"ל החברה הממולאים באופן מקוון.

בשלב מתקדם יותר יבוצע שילוב של Machine Learning כאשר המערכת תחפש ברשתות החברתיות (כגון: Linkadin, Facebook, Twitter etc) ותספק קלט נוסף עבור המועמדים השונים.

בנוסף קיימות פונקציונאליות של הרכבת צוות שיתאים בצורה הטובה ביותר עבור החברה.

שרתי החברה נמצאים ומתוחזקים ע"י AWS והמגייסים השונים (להלן המשתמשים) נגשים למערכת דרך אתר החברה.

בשלב מתקדם המערכת גם תבצע מיפוי של האנשים השונים ברשתות החברתיות ותדע לאתר מחבלים טרם ביצוע זממם.

תוכן

[1. מבוא 2](#_Toc1328107)

[1.1 הגדרת הבעיה 2](#_Toc1328108)

[1.2 ואיך הפרויקט פותר אותה 2](#_Toc1328109)

[1.3 במידה ויש לו מתחרים 2](#_Toc1328110)

[1.4 אולי סיכום קטן על הפרק 3](#_Toc1328111)

[2. רקע תיאורטי 3](#_Toc1328112)

[2.1 רשתות מחשבים 3](#_Toc1328113)

[2.2 מודל client – server 4](#_Toc1328114)

[2.3 ניתוח כלי פיתוח 5](#_Toc1328115)

[2.3.1 סביבת עבודה 5](#_Toc1328116)

[2.3.2 שפות פיתוח 13](#_Toc1328117)

[2.4 מודל הפיתוח 16](#_Toc1328118)

[3. תכנון 16](#_Toc1328119)

[3.1 דרישות המערכת 16](#_Toc1328120)

[3.1.1 דרישות כלליות 16](#_Toc1328121)

[3.1.2 דרישות side client 16](#_Toc1328122)

[3.1.3 דרישות side server 16](#_Toc1328123)

[3.2 אפיון המערכת 16](#_Toc1328124)

[3.3 מבנה המערכת (אופציונאלי – לא בטוח שזה יופיע בדווקא כאן) 16](#_Toc1328125)

[4. תכנון ביצוע 16](#_Toc1328126)

[4.1 תחילת העבודה 16](#_Toc1328127)

[4.2 מודל הפיתוח של המערכת (כאן כדאי להציג מודלים – כדאי לעיין גם בפרויקט של חירום בפיקוד העורף ) 17](#_Toc1328128)

[UI 4.3 17](#_Toc1328129)

[4.4 הקמת שרת Node.js 17](#_Toc1328130)

[4.5 הקמת instance in AWS 17](#_Toc1328131)

[4.6 ארגון בסיס הנתונים (אולי צריך לשלבו עם 4.5) 17](#_Toc1328132)

[Data Mining 4.7 (אולי גם את זה נוסיף פשוט ל4.5) 18](#_Toc1328133)

[4.8 תהליכים 18](#_Toc1328134)

[4.9 פונקציונאליות 24](#_Toc1328135)

[4.10 בדיקות 24](#_Toc1328136)

[5. דיון 24](#_Toc1328137)

[5.1 יתרונות האפליקציה 24](#_Toc1328138)

[5.2 הרחבות אפשריות למערכת 24](#_Toc1328139)

[5.3 אתגרים שהתמודדנו איתם במהלך הפרוייקט 24](#_Toc1328140)

[5.3.1 הכנסת קובץ JSON לFirebase realtime database 24](#_Toc1328141)

[5.3.2 שינוי אורך מחרוזת בSQL 27](#_Toc1328142)

[5.4 רווחים אישיים מהפרויקט 29](#_Toc1328143)

[6. סיכום 29](#_Toc1328144)

[7. ביביליוגרפיה 29](#_Toc1328145)

# מבוא

## 1.1 הגדרת הבעיה

יש בעיה כללית בגיוס עובדים. כשאנשים רוצים לגייס עובד, הם לא תמיד יודעים מהן תכונותיו של המועמד, האם תכונותיו מתאימות למה שהם רוצים או לא. לכן, יש הרבה שיטות לברר מי עובד ומה התכונות שלו.

## 1.2 ואיך הפרויקט פותר אותה

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

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

## 1.3 במידה ויש לו מתחרים

(מי הם ומה ההיתרון שלנו למולם)

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

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

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'.

קישורים[[2]](#footnote-3)

## 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.

קישורים[[3]](#footnote-4)

#### 2.3.1.2 notepad++



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

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

* סימון וסידור תחביר בשפות תכנות שונות
* חיפוש והחלפה עם ביטויים רגולאריים 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

קישורים[[4]](#footnote-5)

#### 2.3.1.3 Tableau

Tableau הוא כלי BI מהמובילים בעולם, עם יכולות ויזואליזציה משמעותיות וממשק משתמש שנחשב נוח ואינטואיטיבי. אין צורך בידע בתכנות בעבודה מול Tableau . הכלי שימושי בתחקור נתונים מעמיק ומורכב, יודע לייצר דוחות ומסייע בקבלת החלטות עסקיות. הכלי תומך ביצירת טבלאות, גרפים ומפות, באופן יחסית נוח וידידותי למשתמש.

קישורים[[5]](#footnote-6)

#### 2.3.1.4 Node.js

Node.js היא פלטפורמה שנבנתה מעל מנוע הג’אווה סקריפט של גוגל -V8 , ומטרתה להוות כלי לפיתוח אפליקציות רשת שעוסקות בטיפול אינטנסיבי במידע, בזמן אמת, ושממעטות בחישובים.

בעזרת Node ניתן להריץ JavaScript ללא צורך בדפדפן. על ידי שימוש בNode ניתן לפתח אפליקציות מבוססות JavaScript למכשירים ניידים ונייחים ולמגוון מערכות הפעלה. Node.js היא שפה גמישה, מונחית אירועים ואסינכרונית המאפשרת גמישות רבה וביצועים מצוינים יחסית למקבילות שלה. Node.js הפכה לפופולרית מאוד בשנים האחרונות.

קישורים[[6]](#footnote-7)

#### 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.

קישורים[[7]](#footnote-8)

#### 2.3.1.6 Amazon Web Services



**Amazon Web Services**  (בראשי תיבות (**AWS** היא יחידה עסקית של חברת אמזון האמריקאית, העוסקת באספקת שירותי מחשוב ענן ליחידים, לחברות ולגופים ממשלתיים ונחשבת לבעלת שירותים מתקדמים בתחום. שירותי הענן של אמזון כוללים מגוון פתרונות תשתית, אחסון, תקשורת, ובסיסי נתונים לצד שירותים ומוצרים משלימים תוך שמירה על רמות אבטחת המידע הגבוהות ביותר.

החברה מספקת סוגים שונים של שירותים בענן המחשוב שלה שהבולטים בהם מתוארים להלן:

1. **Elastic Compute Cloud** - **(EC2)**  השירות מכיל אוסף כלים המאפשרים העלאה של שרתים בגדלים שונים בצורה זריזה ויעילה. ב EC2 ניתן למצוא, לייצר או לרכוש image של שרת המכיל סביבת עבודה נדרשת ולפרוס אותו על מכונה וירטואלית בגודל נבחר ובמיקום נבחר (Instance).
2. **Simple Storage Service S3))** - תשתית אחסון ענן אחסון קבצים סטטיים בקלות ללא הגבלה, וגישה אליהם מכל מקום (ישירות, או באמצעותAPI ) שרותי S3 מתאפיינים בשרידות גבוהה, מחיר נמוך ואפשרויות גדילה כמעט בלתי מוגבלות. הנתונים ב-S3 נשמרים ב,Buckets להם ניתן לשייך הרשאות גישה.
3. **Relational Database Service‏ (RDS)** – תשתית מבוססת ענן למגוון מסדי נתונים, ביניהם MySQL, MariaDBM, SQL server ועוד, מה שהופך את שימוש ב RDS לפשוט. בנוסף קיים שירות מיוחד שמאפשר לעשות העברה של DB קיים אל RDS.

קישורים[[8]](#footnote-9)

#### 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.

קישורים[[9]](#footnote-10)

##### TortoiseGit 2.3.1.7.1



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

##### Github 2.3.1.7.2



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.

קישורים[[10]](#footnote-11)

#### Kanban board 2.3.1.8

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

קישורים[[11]](#footnote-12)

#### Chrome DevTools 2.3.1.9

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.

קישורים[[12]](#footnote-13)

#### Database 2.3.1.10

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[[13]](#footnote-14) , 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.



קישורים[[14]](#footnote-15)

##### PostgreSQL 2.3.1.10.1



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

קישורים[[15]](#footnote-16)

#### 2.3.1.11 JSON

JSON ‏(‎JavaScript Object Notation - ציון אובייקטים ב-JavaScript) הינו פורמט קל משקל להחלפת מידע. פורמט JSON קל לשימוש עבור בני אדם, וקל לפענוח ע"י מכונות. אף על פי שפורמט זה פותח בתחילה לשימוש ב-Javascript ונעשה בו שימוש נרחב בשפה זו, JSON הינו פורמט חופשי לחלוטין משפה, ומשתמש במוסכמות שמוכרות למתכנתים של משפחת שפות C, הכוללת את C, C++, C#, Java, JavaScript, Perl, Python ועוד רבים נוספים. תכונות אלו עושות את JSON לשפה אידיאלית עבור החלפת מידע.

JSON בנוי על שני סוגי מבנים:

* אוסף של זוגות key-value. בשפות השונות, זה בא לידי מימוש כ-אובייקט, רשומה (Record), מבנה (struct), מילון (‎dictionary‏), Hash-table, ‎Keyed list‏ או מערך אסוציאטיבי (Associative array).
* רשימה מסודרת של ערכים. ברוב השפות, זה בא למימוש כמערך (‎Array‏), vector, רשימה (list) או sequence.

אלו הם מבני מידע אוניברסליים. למעשה כל שפות התכנות המודרניות תומכות בהם בצורה זו או אחרת. לכן זה הגיוני ויעיל שמבנה מידע, שהינו בר החלפה עם שפות תכנות, יהיה מבוסס על מבנים אלו.

קישורים[[16]](#footnote-17)

#### 2.3.1.12 HTTP

Hypertext Transfer Protocol הוא פרוטוקול המשמש להעברת קבצי HTML ברשת האינטרנט, הפרוטוקול נמצא בשכבת האפליקציה (במודל ההיברידי ישנם 5 שכבות: פיזית, קו, רשת, תעבורה, אפליקציה)

[HTTP](https://he.wikipedia.org/wiki/%D7%A9%D7%A8%D7%AA_HTTP)  הם שרתי התוכן המרכזיים ברשת האינטרנט ו[דפדפנים](https://he.wikipedia.org/wiki/%D7%93%D7%A4%D7%93%D7%A4%D7%9F) הם תוכנות הלקוח הנפוצות ביותר

כאשר ישנם שני פקודות שכיחות בפרוטוקול, פקודת GET המבקשת דף אינטרנט כלומר מבוצעת בקשת request לserver כדי שיביא לנו דף HTML שביקשנו מה שנקרא response, הפקודה השניה השכיחה היא פקודת POST האחראית להעלאת נתונים לserver.

כפי שהסברנו הפרוטוקול עובד בצורה של בקשה (request) תגובה (response), מה שמכונה ברשתות כפרוטוקול stateless כלומר אינו שומר את המצבים בו הוא נמצא, למשל כאשר יש לנו דף אינטרנט עם מלל ותמונה אזי נדרשות כאן 2 בקשות, אחת עבור המלל והשניה עבור התמונה כאשר הserver לא יודע כי אותו client ביקש את שני הפרטים, בפועל קיים מנגנון שעוקב אחרי הclient והוא cookies .

נתמך ע"י פרוטוקול TCP (בשכבת התעבורה, מספק ריבוב והעברה אמינה של המידע), ישנה אפשרות לעבוד מעל פרוטוקולים לא אמינים כמו UDP כאשר נקרא HTTPU

קישורים[[17]](#footnote-18)

### 2.3.2 שפות פיתוח

#### Javascript 2.3.2.1

JavaScript 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 הודות לפיתוח סביבת ההרצה הנקראת [Node.js](https://nodejs.org/). Node משתמש במנוע להרצת JavaScript שפותח על ידי Google ונקרא V8, ובעזרתו ניתן להריץ JavaScript ללא צורך בדפדפן. על ידי שימוש בNode ניתן לפתח אפליקציות מבוססות JavaScript למכשירים ניידים ונייחים ולמגוון מערכות הפעלה.

קישורים[[18]](#footnote-19)

#### 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.

קישורים[[19]](#footnote-20)

#### 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.

#### Python 2.3.2.4

היא שפה עילית המפותחת בקוד פתוח , שפת תכנות דינמית 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 אשר יוכל לטפל בחריגות עבורAmoeba OS, 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.

אנחנו השתמשנו בגירסת Python 2.7 במהלך הפרוייקט.

קישורים[[20]](#footnote-21)

## 2.4 מודל הפיתוח

מרכיב מרכזי ביותר בניהול פרויקטי תוכנה הוא הגדרה מסודרת של מחזור חיי הפרויקט. פרויקט הוא רצף של צעדים שתחילתו בייזום וסופו בהתקנה ומסירה ללקוח. באמצע הרצף מצויים אפיון מסודר, עיצוב, בנייה, בדיקות ועוד בשיטות פיתוח שונות.

שרשרת זו מכונה מחזור חיי הפרויקט. כל מתודולוגיה מכובדת לניהול פרויקט מציגה את מחזור החיים שלה, כאשר כולם מדברים, פחות או יותר, על אותו מחזור חיים, או לפחות על המרכיבים הבסיסיים שלו. לעומת זאת, קיימות מספר שיטות פיתוח למימוש המערכת. שיטות אלה שונות ונבדלות האחת מהשנייה. השיטות המקובלות בתעשייה הן:

* [השיטה הסדרתית](http://www.methodacloud.com/content/pages/kit_agile/H_Guide.htm#_%D7%94%D7%A9%D7%99%D7%98%D7%94_%D7%94%D7%A1%D7%93%D7%A8%D7%AA%D7%99%D7%AA_(%D7%A1%D7%91%D7%91)
* [השיטה האיטרטיבית](http://www.methodacloud.com/content/pages/kit_agile/H_Guide.htm#_%D7%94%D7%A9%D7%99%D7%98%D7%94_%D7%94%D7%90%D7%99%D7%98%D7%A8%D7%98%D7%99%D7%91%D7%99%D7%AA,_%D7%99%D7%97%D7%99%D7%93%D7%AA)
* [מתודולוגיות זריזות](http://www.methodacloud.com/content/pages/kit_agile/H_Guide.htm#_%D7%94%D7%A9%D7%99%D7%98%D7%94_%D7%94%D7%90%D7%99%D7%98%D7%A8%D7%98%D7%99%D7%91%D7%99%D7%AA,_%D7%9E%D7%A1%D7%A4%D7%A8)

מודל השיטה הסדרתית (נקראת גם מודל מפל המים water fall), הוא מודל ותיק ורווח (בלשון עבר) בקרב התעשייה,

ע"פ שיטת מודל מפל המים מחזורי הפיתוח שונים, כלומר הפרויקט כולו מורכב מכמה חלקים כאשר בכל פעם מתמקדים באותו החלק עד לסיומו בצורה מקצועית וממצה ורק לאחר שסיימנו באופן מקיף לסיים את השלב, נוכל להמשיך לשלב הבא (כאשר אין דרך חזרה כמו מפל מים שבו המים רק מתקדמים מטה ולא חוזרים אחורה).

מתודולוגיית מפל המים שמה דגש רב על איסוף וניתוח של כל הדרישות כולן קודם לתחילת הפיתוח, וממליצה שתהליך הפיתוח לא יחזור לאחור לאחר ששלב מסוים בו הסתיים. השלבים העיקריים בשיטה זו הם [איסוף וניתוח דרישות](https://he.wikipedia.org/w/index.php?title=%D7%90%D7%99%D7%A1%D7%95%D7%A3_%D7%95%D7%A0%D7%99%D7%AA%D7%95%D7%97_%D7%93%D7%A8%D7%99%D7%A9%D7%95%D7%AA&action=edit&redlink=1), [עיצוב תוכנה](https://he.wikipedia.org/wiki/%D7%A2%D7%99%D7%A6%D7%95%D7%91_%D7%AA%D7%95%D7%9B%D7%A0%D7%94), [תכנות](https://he.wikipedia.org/wiki/%D7%AA%D7%9B%D7%A0%D7%95%D7%AA), [בדיקות](https://he.wikipedia.org/wiki/%D7%91%D7%93%D7%99%D7%A7%D7%95%D7%AA_%D7%AA%D7%95%D7%9B%D7%A0%D7%94), שילוב, [התקנה](https://he.wikipedia.org/w/index.php?title=%D7%94%D7%AA%D7%A7%D7%A0%D7%94&action=edit&redlink=1) ו[תחזוקה](https://he.wikipedia.org/wiki/%D7%AA%D7%97%D7%96%D7%95%D7%A7%D7%AA_%D7%AA%D7%95%D7%9B%D7%A0%D7%94).

### 

### השיטה האיטרטיבית, יחידת מסירה/גרסה אחת

בשנות התשעים החלה להופיע שיטת הפיתוח בסבבים, או בשמה האחר, השיטה המדורגת או האיטרטיבית (Iterative Development). מדובר בפיתוח מרובה יחידות מסירה תוך חלוקת המערכת לתת-מערכות וכתוצאה מכך חלוקת הפרויקט לתת-פרויקטים. שיטה זו מבוססת על הרעיון המקובל בכלל התעשייה של בניית מוצר בגירסאות שונות.

### פיתוח זריז, מספר יחידות מסירה/גרסאות

אג׳ייל היא מתודולוגית פיתוח השואפת להתייחס לפיתוח תוכנה בצורה שונה מבחינת זריזות וגמישות התהליך.

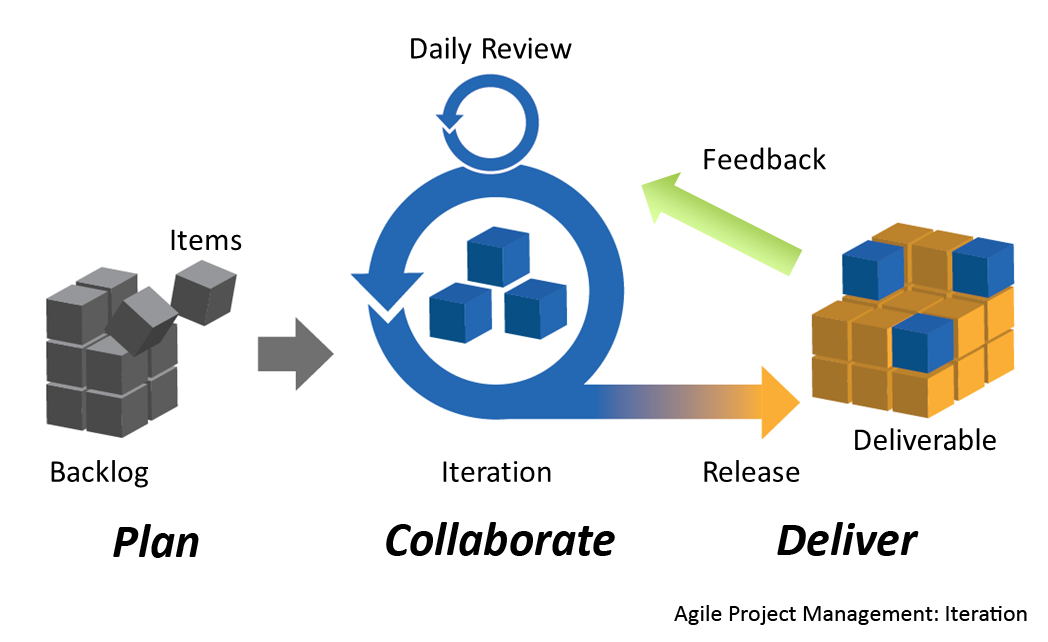
כפי שהסברנו על מודל מפל המים בה אנו מתחילים באפיון מקיף של הפרוייקט כדי להבין את כל המשמעויות והנגזרות הטכניות שלו. אח״כ אנחנו מחלקים את המשימות בין צוותי פיתוח שונים כל חלק בתהליך הפיתוח תלוי בחלק שלפניו, כך שלא ניתן למשל להתחיל בפיתוח הפונקציונליות לפני שעיצוב הפרוייקט מוכן, מאושר וסגור באופן סופי.

במתודולוגית אג׳ייל הנקראת לפעמים גם **סקראם SCRUM** הגישה לפרוייקט היא לפרק אותו לספרינטים מינימליים אותם נחלק בין צוותי פיתוח המסוגלים לסגור כל ספרינט שכזה מקצה לקצה. כך לדוגמה במקום לאפיין את כל האתר, ואז לעצב אותו וכ׳ו, אנו יכולים לאפיין את דף הבית בלבד, לעצב אותו ולהעלות אותו לאוויר ובשלב ב׳ להוסיף עמודים פנימיים, או מודולים ופיצ׳רים נוספים שלא היו באפיון המקורי שלו.

בדרך זו אנו יכולים לספק אתר פעיל בזמן קצר מאוד ולהבטיח שביעות רצון גבוהה יותר לאורך כל התהליך.

**עקרונות שיטת Agile**

* ראש סדר העדיפויות הוא לספק מוקדם ובאופן רציף ככל האפשר תוכנה בעלת ערך ללקוח.
* מצפים לשינויים בדרישות התוכנה, אפילו בשלב מתקדם של הפיתוח. תהליכים מהירים ברי-שינוי מאפשרים יתרון משמעותי בתחרות הלקוח.
* אספקה של תוכנה עובדת באופן תדיר, ממספר שבועות עד למספר חודשים, עם שאיפה לסולם זמנים קצר.
* אנשי פיתוח והמשתמש העסקי חייבים לעבוד ביחד באופן יום יומי לאורך הפרויקט.
* הדרך האפקטיבית והמספקת ביותר לשיתוף מידע בקבוצת הפיתוח היא שיחה פנים מול פנים.
* תוכנה עובדת היא יחידת המידה העיקרית להתקדמות אמיתית בפיתוח.
* המשקיעים, המפתחים והמשתמשים צריכים לשמור על קצב קבוע לאורך זמן.
* יש לספק תשומת לב מתמשכת למצוינות טכנולוגית ועיצוב (Design) טוב המאפשרים גמישות
* פשטות – היא חיונית (האומנות למקסם את כמות העבודה שלא תעשה).
* הארכיטקטורה, העיצוב והדרישות חייבות להתבצע בתוך "קבוצות עצמאיות".
* בזמנים רגועים, על הצוות להגות ולחשוב על איך להפוך להיות יותר אפקטיבי, מאשר לדמות את התנהגות שלו לזה.



<https://he.wikipedia.org/wiki/%D7%9E%D7%AA%D7%95%D7%93%D7%95%D7%9C%D7%95%D7%92%D7%99%D7%99%D7%AA_%D7%A4%D7%99%D7%AA%D7%95%D7%97_%D7%AA%D7%95%D7%9B%D7%A0%D7%94#Scrum>

<http://www.methodacloud.com/content/pages/kit_agile/H_Guide.htm>

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

[שיטת העבודה שלנו]

# תכנון

## 3.1 דרישות המערכת

חלק ראשון היה נדרש ליצור DB שיכיל הרבה תפקידים שונים ומגוונים וסל תכונות עבור התפקידים השונים במינונים שונים.

החלק השני היה אחראי על Data mining, כלומר איסוף מידע ספיציפי מאתר גדול ולשלבו בתוך הDB הקיים לנו (כאמור משלב קודם)

החלק השלישי אמון על הרכבת צוותים באופן כזה שנוכל לנבא בקירוב טוב את ההרכב שיוכל להתאים בצורה הטובה ביותר למשימות שהחברה מייעדת להם.

### 3.1.1 דרישות כלליות

מטרת המערכת היא לאפשר למשאבי אנוש ולמגייסים השונים, ובחלק העתידי שלה גם לעזור לכוחות הביטחון לאסוף מידע בצורה חכמה מקצועית ומדויקת על מועמדים לתפקידים השונים בחברה, תוך חיסכון בזמן, משאבים וכסף.

למעשה מתקבלת תוצאה של קבלת מועמדים מתאימים יותר עבור התפקיד וגם הרכבת צוותים יותר מקצועיים

### 3.1.2 דרישות side client

מטרת צד הלקוח היא לאפשר ממשק משתמש נוח וידידותי.

השימוש במערכת יהיה פשוט וקל, ללא צורך בידע או בכישורים מוקדמים, המבנה החזותי של המערכת יהיה נוח, ברור, נקי ולא עמוס בכפתורים, ונעים לעין. ללא מלל רב וללא צורך בהקלדה רבה. ביצוע הפעולות יהיה נעים ונח. הגישה לנתונים תהיה מהירה.

למשתמש המערכת תיהיה שליטה ע"י dashboard ניהולי ועל ידו יוכל לקבוע אם הוא מעוניין לקבל את המלצות המערכת (המובססות על שנים רבות של מחקר) או לחילופין לשנות את "המינונים" השונים עבור תכונות מסוימות לתפקיד כלשהו בחברה על סמך הניסיון האישי שלו.

לסיכום, ניתוח הדרישות הינו שלב הכרחי להבנת דרישות הלקוח במדויק. חשוב להשקיע את המשאבים והזמן הנחוצים לשלב זה מאחר והוא מהווה בסיס לשלבים הבאים בפרויקט, ומשפיע רבות על תוצר הפרויקט הסופי.

### 3.1.3 דרישות side server

מטרת צד השרת היא ניהול התפקידים השונים והגדרות מינוני התכונות עבור התפקידים.

המערכת תאפשר ניהול מאגר הישויות על מאפייניהם, ביצוע משימות בקרה על הישויות, הפקת מידע המצביע על המועמדים המתאימים ביותר לתפקיד.

דרישות הלקוח עבור צד השרת הן:

1 . המערכת תגדיר את הישויות השונות שהן - ארגון, תפקיד, תכונה, מועמד, (שאלון). עבור אותם התפקידים

משתמשי המערכת, מערכת ההרשאות מרבדת את מאפייני הפעילות לפי סוג

המשתמש. המשתמשים מונים עובדי אגף הביטחון תחזוקה, מנהלי שטח וראש אגף -

ביטחון של נמל אשדוד.

2 . המערכת תאפשר ביצוע בדיקות עבור מועמד ומידת התאמתו לתפקיד המוצע, עבור כל תפקיד מוגדרות

תכונות אישיות רצויות תפקיד כאשר כל תכונה מחולקת לתת תכונה, כאשר ישנם חמישה תתי תכונות עבור כל תכונה.

החלוקה לתתי תכונות מאפשרת לבצע הגדרה מדויקת יותר עבור המינון הרצוי לתפקיד המוצע, כך למשל אם נרצה לגייס עובד לתפקיד שיווק ונמדוד את תכונת הפתיחות אנחנו נחלק אותה לחמישה חלקים הבאים:

פתיחות נמוכה, פתיחות מתחת לממוצע, פתיחות ממוצעת, פתיחות מעל לממוצע ופתיחות גבוהה. כמובן שעבור תפקיד שיווקי נרצה פתיחות ברמה גבוהה, ולכן נסמן את "פתיחות גבוהה" כרצויה מאוד. את שאר האפשרויות של פתיחות נסמן כלא רצויות. וכך נעבור עבור כל "סל" התכונות ששייכות לתפקיד ונבדוק את מידת התאמת המועמד ותכונתיו למול המלצת המערכת או לחילופין הגדרת המשתמש בצורה ידנית ע"פ שיקול דעתו

3 .המערכת תאפשר להרכיב צוותים המתאימים ביותר בתחומם (על פי מחקרים). המערכת תאפשר לייצר תמהיל של אנשי צוות שהתכונות שלהם (לפעמים דווקא תכונות שונות), מאפשרות את ביצוע תפקידו של הצוות בצורה אופטימלית.

## 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

בתחילה לאחר מפגש ארוך עם מנכ"ל החברה נדרשנו ליצור

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 ) - כבר קיים בסעיף 2.4

## 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 Mining 4.7 (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, וכך עשינו.

משימה נוספת שקיבלנו היתה לייצר כלי שיהי' בסיס למערכת ליצירת צוותים. החברה מעוניינת לאפשר למשתמשים לנסות להרכיב באופן אוטומטי צוותים אידאליים. לצורך זה, התבקשנו לייצר מודל חישובי שיאפשר למיין את המועמדים לפי ציון משוקלל, ולבחור את הטובים ביותר. הציון המשוקלל היה צריך להתבסס על חמש תכונות OCAEN (openness, consciousness, extraversion, agreeableness, neuroticism) שנחשבות לתכונות המרכזיות על פי מחקרים. לאחר שיחות עם מנכ"ל החברה, הגענו למודל הבא –

1. כל אחת מחמשת התכונות מקבלת ציון שמגדיר את המצב האידאלי. (למשל, openness 5 פירושו שהמצב האידאלי עבור openness הוא הציון 5. הציונים הם בטווח 1-5)
2. כל אחת מהתכונות מקבלת משקל מסויים, שמייצג את החשיבות שלה ביחס לציון הסופי. סך כל המשקלים מגיע ל-1.
3. נבדקים הציונים של המועמד עבור כל אחת מהתכונות.
4. עבור כל תכונה, מחושב הפער בין הציונים האידאליים לציון של המועמד. כמו כן, מחושבת השונות של הפערים.
5. הפער מוכפל במשקל של התכונה הספציפית, ובשונות של כל הפערים, ומתקבל הציון הסופי.
6. המועמדים ממויינים ברשימה לפי הציונים שלהם.
7. מתוך הרשימה נבחרות קבוצות בגודל המתאים (כאשר הקבוצה הראשונה, שהגיעה מראש הרשימה, היא הקבוצה הטובה ביותר)

## 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

חברות המייצרות מוצרים, נדרשות לוודא שכל גירסה עבור המוצר שהן משווקות, עומד בתקנים

ומספק את כל features שהוא אמור לתת. בדיקות תוכנה (QA)מהוות חלק חשוב בקביעת מדד איכות המוצר.

מטרת הבדיקות, לאמת את אמינות התוצרים ונכונותם, לבדוק את מידת התאמתם לדרישות. כמו

כן, תהליך הבדיקות מאפשר איתור שגיאות בשלבים מוקדמים ובכך לצמצם עלויות. ולרכוש אמינות אצל הלקוחות.

כחלק מהפרויקט התבצעו בדיקות רבות, כאשר את הבדיקות בצד הלקוח ביצענו על קובץ HTML שיצרנו והבדיקות בצד השרת ביצענו ע"י בדיקת log-ים בNode js command prompt וע"י pgAdmin כדי לתקשר עם הinstance הקיים לנו בAWS .

נפרט דוגמה

# 5. דיון

## 5.1 יתרונות האפליקציה

עד כה אנחנו נוכחים לראות את הקושי הרב בגיוס עובדים חדשים לארגון.

בתהליך מעורבים: HR (משאבי אנוש), TL (ראש צוות), ומנהל, ישנם ארגונים בהם עוברים תהליכים נוספים כמו מעבר דרך מכונים פסיכוטכניים וכו'.

בחלוקה גסה מדובר על אנשים שזוהי הגדרת תפקידם (לאתר מועמדים למשרה), מכונים טכניים שזהו עיקר עיסוקם (וכאן נכנס שיקול כלכלי) ואנשי צוות הנותנים ערך לחברה (הכוונה באופן ישיר למוצר החברה)

ע"פ החלוקה הנ"ל קל לראות שמדובר בשרשת של אנשים עד לגיוס המיוחל.

כאשר אנחנו מפנים טווח גדול יחסית של מועמדים לTL אזי שאנחנו מבזבזים את זמנו לריק – מדובר כאן בדבר משמעותי כיוון שבאותו הזמן הוא נמנע מלהמשיך ולבצע את משימותיו השותפות, גם ניתן לומר זאת עבור המנהל שמעליו – אומנם הוא לא יזקק לביטול זמן גדול כיוון שאותו מועמד כבר עבר כמה סינונים אך עדיין יכול להיפסל בשלב זה.

כך גם לגבי המכונים, כאן מדובר בהפסד כלכלי כאשר שולחים מועמדים שיכלנו לנבא את הצלחתם בשלב מוקדם יותר.

התהליך כשלעצמו מייגע ולפעמים נוכל אכזבות וכשלונות, אכזבות – כיוון שלאחר כל המאמצים שהושקעו הוא נפל בשלב מאוד מתקדם או לחילופין החליט לסרב להצעה (לאחר קבלתו לחברה)

כשלונות – כיוון שיכול להיות שבסופו של דבר קיבלנו עובד שאולי מגיע עם נתונים טובים אך אינו מתאים דיו למשרה ואת זה נגלה רק לאחר תחילת תפקודו במשרה.

מתוך כל החושך האופף את שיטה הנוכחית לגיוס עובד, החברה Cyber Traits מעוניינת להביא בשורה חדשה ומבטיחה לתחום.

האפליקצייה (במעמד הנוכחי) יכולה לתת מענה טוב יותר לארגונים ע"י כך שהמומעד צריך להיכנס ולמלא טפסים/שאלונים.

המערכת מגיעה עם DB המכיל את כל התפקידים, כאשר עבור כל תפקיד מתקבל "סל" של תכונות, לרוב מוגדרות מראש ע"פ מחקריו של מנכ"ל החברה וע"פ נתונים שייאספו עוד מהשטח כדי לקבל רמת דיוק גבוהה ככל האפשר, יחד עם זאת ניתן האפשרות לאנשי הHR לשנות את אותן ההגדרות כפי ראות עינייהם.

לאחר עיבוד הנתונים של המועמדים למול המשרה המוצעת מתקבלת תשובה עם המועמדים הכי מתאימים למשרה.

כבר אנחנו יכולים להבין כי נדרשת פחות השקעה אפילו מצד אנשי HR בגיוס הקיים היום, ועוד יותר חשוב אנחנו מורידים משמעותית את כמות המועמדים שימשיכו לשלבים הבאים (בו כמובן מעורבים, TL,מנהל, מכונים וכו'), כלומר מתקבל חיסכון משמעותי בזמן, כסף ושימוש במשאבים (ביטול זמן של TL למשל)

ערך נוסף המתקבל מהאפליקצייה הוא האפשרות להרכיב צוות שיהיה המתאים ביותר עבור משימה שהחברה הגדירה, לא פעם מתקבלת החלטה בחברה לייצר מוצר חדש ומכאן עולים הלבטים אילו אנשים ירכיבו את הצוות, בדר"כ אנשי הצוות מגיעים מתוך החברה ולכן זה יותר קל אבל מצד שני איך נדע את מי להרכיב? מה גם שלפעמים הרכבת צוות לא נכון יכול לגרור כישלון של הפרויקט ולהפסדים כלכליים גדולים לחברה.

הפתרון המוצע נותן תחזית טובה יותר עבור הרכבת הצוות המתאים ביותר עבור המשימה.

## 5.2 הרחבות אפשריות למערכת

5.2.1 שילוב Machine Learning

5.2.2הרחבה אפשרית היא היכולת לוותר באופן מלא או חלקי על השאלונים הממולאים באופן ידני על המועמד, כמובן שמדובר בשאלונים מבוררים היטב ע"י פסיכולוגים כדי להקשות על המומעד "לשקר" בתשובותיו, גם בשלב זה אנחנו נשארים במעט חוסר ביטחון מלא ביכולותיו ותכונות אישיותיו של המועמד, מה שאין כן כאשר אנחנו ננצל את היכולת להכניס למערכת כinput את כל הפרופילים האישיים שלו ברשתות החברתיות השונות, כגון: Linkadin, Facebook וכו', ע"י אלגוריתמים חכמים של Machine Learning, מתוך נק' מוצא כי המועמד שמנהל את הפרופיל שלו כמה שנים כבר לא יכול לשקר (נוכל גם לבצע השוואה בין הרשתות החברתיות השונות), וכך נקבל אומדן טוב יותר.

איתור מחבלים ופושעים טרם ביצוע זממם

גם כאן נוכל לאפיין ע"י שימוש ב Machine Learning על פרופילים של אנשים (כאן בשונה מהסעיף הקודם נוכל לקבל אישורים מגופי הביטחון להיכנס לפרופיל גם כאשר לא קיבלנו הרשאה לכך), וע"י המידע הקיים וכמובן בתוספת הניסיון שנצבר ע"י Machine Learning אנחנו נוכל לאפיין חשודים לפני ביצוע פעולת טרור או עוון פלילי ובכך לסכל פיגועים ולמנוע אסונות.

## 5.3 אתגרים שהתמודדנו איתם במהלך הפרוייקט

### 5.3.1 הכנסת קובץ JSON לFirebase realtime database

כזכור, במהלך העבודה על איסוף הנתונים מonetcenter, העלינו אותם לfirebase. כדי להעלות את הנתונים בנינו קובץ JSON גדול ורצינו לעשות לו import לתוך firebase, אבל קרתה תקלה



כלומר יש תווים בקובץ שלנו שבעייתיים עבור firebase במקרה שלנו בתו (.) רצינו לטפל בזה באופן פשוט ע"י האפשרות של replace שנמצא בnotepad++ אבל בשביל לעשות את זה היינו צריכים לאפיין את כל המקרים בהם זה מופיע ולכן נדרשנו להשתמש בביטוי רגולרי וניסינו את הביטוי הבא

-\d\d\d\d\x2e ולהחליפו ב -\d\d\d\d\_נתקלנו בבעיה שלא שמרנו על המפתחות אלא החלפנו גם את הערכים המספריים שלהם למחרוזת dddd . חיפשנו איך לשמר את הערכים המקוריים שהתקבלו ע"י הביטוי הרגולרי שלנו ומצאנו שאפשר לשמר אותם בצורה הבאה –(\d\d\d\d)\x2e להפוך ל -\1\_ שלמעשה 1\ היא הקבוצה שנמצאת בסוגריים וכך שמרנו על הערכים הללו







<https://stackoverflow.com/questions/14458160/while-replacing-using-regex-how-to-keep-a-part-of-matched-string>

למרות השינויים שביצענו עדיין קיבלנו את עמוד השגיאה של הFB.

חיפשנו את הבעיה באינטרנט ומצאנו בstackOverFlow תשובה שלפיה צריך שיהיה קידוד UTF-8 בדווקא ולא קידוד UTF-8-BOM שהקובץ שלנו היה מקודד לפיו .

לכן שנינו את הקידוד באופן הבא



<https://stackoverflow.com/questions/38668477/firebase-invalid-json-data-or-key-values-key-values-can-t-contain-o>

לאחר השינוי ראינו שאכן הצלחנו לבצע import לFB



### 5.3.2 שינוי אורך מחרוזת בSQL

בסיום העבודה על איסוף הנתונים מonetcenter, הכנסנו את הנתונים לתוך טבלה בSQL.

הצלחנו לייצר טבלה כאשר העמודה הראשונה הוגדרה להיות מסוג char באורך של 30 תווים והכילה את הקוד של כל מקצוע

הטבלה הכילה את כל המידע שעיבדנו מהאתר O\*Net ע"י סקריפט שכתבנו בpython ותפקידו היה גם להכניס את הנתונים לאחר עיבודים לתוך הpostgres

לאחר מכן לקחנו טבלה אחרת שמקשרת בין קוד של מקצוע ושם של מקצוע מתוך האתר ועיבדנו גם אותה ורצינו לשנות את העמודה הראשונה מקוד לשם מקצוע. הבעיה בה נתקלנו היא שישנם שמות ארוכים יותר מ30 תווים וניסינו ע"י חיפוש באינטרנט לשנות בטבלה בpostgres את כמות התווים ולהגדילה ל100 תווים כך שבטוח יכנסו כלל השמות האפשריים.

אך נתקלנו בשגיאה בכל מיני ניסיונות



הלכנו לחפש באינטרנט מהי השגיאה וגם את מס' הstate אך לא קיבלנו תשובות שקידמו אותנו לתשובה

ולכן היינו צריכים לחפש כמה פעמיים ובעוד מקומות עד שהגענו לתוצאה המבוקשת

<http://www.firebirdfaq.org/faq285/>



וכאן ניתן לראות את השינוי



הרצנו שוב את הסקריפט וקיבלנו exception שעדיין 100 תווים לא מספיקים ולכן הגדלנו ל150.

לאחר הרצת הסקריפט קיבלנו את התוצאה הבאה



974 שורות בטבלה

## 5.4 רווחים אישיים מהפרויקט

בזמן העבודה על הפרויקט, עברנו תהליך שלם של גילוי ולמידה. העבודה הייתה

עצמאית ודרשה מאתנו להשתמש בידע שנרכש בקורסים שונים לאורך הלימודים ושילובם זה בזה, יחד עם זאת נדרשנו ולרכוש ידע חדש תוך כדי הפיתוח ולהגיע לכדי יצירה מוגמרת העונה על כלל דרישות הפרויקט.

רכשנו ידע רב וכלים רבים בתחום פיתוח WEB & Cloud תחום מבוקש שהולך ותופס תאוצה עם השנים, צברנו ניסיון רב שעתיד לתרום לנו בהמשך הדרך.

בשלבי התכנון וכן במהלך העבודה, למדנו רבות על ארכיטקטורת הפיתוח ועל המודלים השונים איתם עבדנו.

המערכת פותחה באמצעות טכלונוגיות שונות, חלקן הכרנו ונדרשנו להעמיק את ההכרות

איתן וחלקן היו חדשות עבורנו, בתחילת העבודה על הפרויקט כחלק מההכנות לקראתו,

למדנו אותן, וכן במשך הפרויקט כשהיה צורך להשתמש בטכנולוגיות חדשות נוספות. בכל

שלב עברנו על הקוד שכתבנו וביצענו שיפורים ותיקונים על מנת להגיע למקסימום יעילות.

הלמידה הראשונית לאחר ההכרות עם השפות השונות וסביבת הפיתוח, הייתה כיצד

להעביר את דרישות החברה מתיאוריה למעשה. ניתוח דרישותיה, תחקור חוזר ונשנה אודות הפרטים הקטנים והמכלול ותרגומן לשפת הקוד.

למדנו כיצד לחפש פתרונות לבעיות במרחב האינטרנטי[[21]](#footnote-22) ומתי וכיצד למצוא פתרון יצירתי עצמאי.

פעמים רבות חשבנו על מספר דרכי פתרון עבור משימה או בעיה שעלתה ונדרשנו לנהל

דיון מושכל עם עצמנו בנוגע ליתרונות ולחסרונות שבכל דרך, ולבצע בחירה מתאימה, על אף שבפעמים מסוימות מנכ"ל החברה הכריע על דרך מסוימת.

העבודה על הפרויקט דרשה מאתנו כישורים נוספים כגון ניהול וחלוקה נכונה של זמן,

חשיבה מחוץ לקופסא.

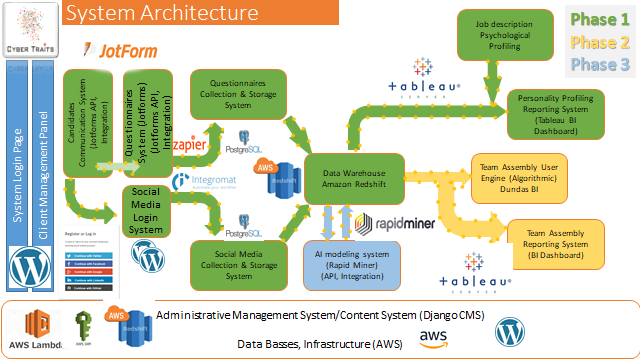
בתהליך שעברנו קיבלנו מושגים על אופן העבודה בשוק ועל ההתנהלות בתעשיית ההייטק ובפיתוח מיזמים מהשלב הרעיוני ועד לתוצר הסופי.

למדנו להעמיד את עצמנו בנעליהם של הלקוחות הפוטנציאלים, בכל שלב ערכנו תשאול לו

הינו המשתמש איך הינו רוצים שזה יראה, מה תיהיה הפונקציונאליות, אילו דברים חסרים, אלו דברים מיותרים וכו.

# 6. סיכום

# 7. ביביליוגרפיה





1. כיוון שכדי שהclient יוכל ליצור קשר למול הserver שלנו הוא נדרש לדעת לאיזה port לגשת [↑](#footnote-ref-2)
2. <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/> [↑](#footnote-ref-3)
3. <https://confluence.jetbrains.com/display/PYH/Getting+Started+with+PyCharm> [↑](#footnote-ref-4)
4. <https://he.wikipedia.org/wiki/Notepad%2B%2B> [↑](#footnote-ref-5)
5. <http://www.sitqad.co.il/%D7%90%D7%A0%D7%9C%D7%99%D7%A1%D7%98-%D7%9E%D7%A0%D7%94%D7%9C-bi-%D7%9E%D7%97%D7%A4%D7%A9-%D7%9C%D7%A2%D7%A0%D7%95%D7%AA-%D7%A2%D7%9C-%D7%A9%D7%90%D7%9C%D7%95%D7%AA-%D7%91%D7%9E%D7%94%D7%99%D7%A8/>

   <https://www.naya-college.co.il/%D7%9E%D7%94-%D7%96%D7%94-tableau/> [↑](#footnote-ref-6)
6. <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/>

   <https://internet-israel.com/category/%D7%9E%D7%93%D7%A8%D7%99%D7%9B%D7%99%D7%9D/node-js-%D7%9E%D7%93%D7%A8%D7%99%D7%9B%D7%99%D7%9D/> [↑](#footnote-ref-7)
7. <https://en.wikipedia.org/wiki/Firebase>

   <http://meta-pa.blogspot.com/2017/08/firebase.html> [↑](#footnote-ref-8)
8. <https://www.israelclouds.com/glossary>

   <https://aws.amazon.com/rds/>

   <https://he.wikipedia.org/wiki/Amazon_Web_Services>

   <https://dapps.co.il/%D7%A9%D7%99%D7%A8%D7%95%D7%AA%D7%99-%D7%94%D7%A2%D7%A0%D7%9F-%D7%A9%D7%9C-amazon-%D7%A9%D7%90%D7%AA%D7%9D-%D7%97%D7%99%D7%99%D7%91%D7%99%D7%9D-%D7%9C%D7%94%D7%9B%D7%99%D7%A8-%D7%95%D7%93%D7%A8%D7%9B/>

   <https://www.matrix.co.il/About/Pages/amazon.aspx> [↑](#footnote-ref-9)
9. <https://www.mrcoral.co.il/%D7%9E%D7%94-%D7%96%D7%94-git-%D7%92%D7%99%D7%98/> [↑](#footnote-ref-10)
10. <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/> [↑](#footnote-ref-11)
11. <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)> [↑](#footnote-ref-12)
12. <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> [↑](#footnote-ref-13)
13. כמובן בהנחה שלא מתייחסים לחישוב החודשים אלא למס' עגול המבטא את גילו של התלמיד [↑](#footnote-ref-14)
14. <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> [↑](#footnote-ref-15)
15. <https://www.postgresql.org/about/> [↑](#footnote-ref-16)
16. <http://www.json.org/json-he.html>

    <https://he.wikipedia.org/wiki/JSON> [↑](#footnote-ref-17)
17. <https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol>

    <https://he.wikipedia.org/wiki/Hypertext_Transfer_Protocol> [↑](#footnote-ref-18)
18. <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/> [↑](#footnote-ref-19)
19. <https://en.wikipedia.org/wiki/HTML>

    <https://www.w3schools.com/html/html_intro.asp> [↑](#footnote-ref-20)
20. <https://he.wikipedia.org/wiki/%D7%A4%D7%99%D7%99%D7%AA%D7%95%D7%9F> [↑](#footnote-ref-21)
21. Stackoverflow, Stackexchange etc. [↑](#footnote-ref-22)