21/11/2017

**שלבי הפתרון:**

1. יצירת בסיס הנתונים ב-mySql.
2. כתיבת צד שרת REST API ב-php, לצורך ביצוע פעולות CRUD מול בסיס הנתונים (יצירה, קריאה, עדכון, ומחיקה). בנוסף, כתיבת ממשק לפעולת חיפוש לפי שם פרטי של סטודנט.
3. כתיבת התוכנה לצד הלקוח ב-AngularJS Material.

**MySQL**

* שמות טבלאות ביחיד ולא ברבים, חלק מההמלצות. עניין של בחירה ועקביות. ניתן לעבוד כפי שמורגלים בחברה, לא קריטי.
* עבור שדות המפתח עיר, מספר ת.ז – תמיכה עד מעל 4 מיליארד רשומות (10 ספרות עבור ת.ז, מספיק לבינתיים למספר ת.ז ישראלית).
* לגבי הבחירה בשדה tinyint(1) עבור שדה כן/לא - עד לגרסה 5.0.3 של mySQL bit(1) היה שקול ל tinyint(1). אבל מגרסה זו – bit(1) אכן יאפשר שמירה של 0 ו-1 בלבד. בכל מקרה, tinyint(1) מאפשר שמירה של הספרות 0-9, כאשר רק 0 הוא false וכל השאר הם true.
* עבדתי בפיתוח עם גרסת Workbench 6.3 ועם גרסת mySQL 5.5.45
* הנתונים של בסיס הנתונים נשמרים בתיקיית programData, בתצורת iNNoDB לכל טבלה קובץ frm משלה שמגדיר את הנתונים והמבנה. בנוסף ישנו קובץ ibdata1 שמכיל נתונים עבור תחזוקת הטבלאות.

**php**

* בקובץ **database** מגדירים מחלקה שתפקידה להחזיק את החיבור (connection) לבסיס הנתונים. כל הפרמטרים לחיבור נמצאים אך ורק בקובץ הזה.
* בקובץ **database**, מוסיפים את השורה

$this->conn->exec("set names utf8");

שתפקידה לאפשר קידוד של utf8 בשאילתות. מניח שיש אלטרנטיבות לכך.

* נעשה שימוש ב-**PDO**, שהוא רכיב לממשק משותף לכמה דרייברים שונים לחיבור לבסיס נתונים ולאו דווקא ל mySQL, כמו האופציות האחרות שהן mySQL הישנה יותר ו mySQLi החדשה והמשופרת. mySQLi מציעה שתי צורות שונות – גישה פרוצדורלית וגישה מונחית-עצמים. דוגמה לפרוצדורלית:

$mysqli = mysqli\_connect( 'localhost', 'username', 'password', 'database');

ודוגמה למונחית עצמים:

$mysqli = new mysqli( 'localhost', 'username', 'password', 'database');

יתרון מרכזי של PDO – תומך ב-12 סוגי שונים של בסיסי נתונים, בניגוד ל mySQL שתומך רק באחד.

* **Prepared Statements** – לצורך מניעת מצב של SQL Injection, משתמשים בשאילתה מוכנה כדי לקבוע את פרמטרים הדינמיים שיעברו. בשלב הבא, ניתן לעשות binding לפרמטרים בשתי שיטות – או לפי מספר הפרמטר או לפי השם שלו.
* שליפת הרשומות מבסיס הנתונים אפשרית במספר שיטות. בהשוואה בין fetch מול fetchAll, קראתי בכמה מאמרים שעדיף מבחינת ביצועים לשלוף עם fetch שורה אחת בכל פעם ולהשתמש בתוך לולאת while, וגם מבחינת השימוש בזיכרון, במקרה ויש הרבה נתונים, אז fetchAll יגזול הרבה זיכרון.

**angularJS**

* לצורך עבודה מול ה-api, נעשה שימוש ב-**factory** ולא ב-**service**. Service הוא בעצם constructor function שבעצם נקרא מאחורי הקלעים בעזרת המילה new, נעשה בו שימוש במילה this, צריכים לאתחל אותו ב-constructor ומוחזר המופע של ה-service. בכל מקרה, מאחורי הקלעים באנגולר נעשה תמיד שימוש ב-factory, גם אם השתמשנו ב-service. הבדל מרכזי ביניהם: **factory מחזיר אובייקט**. היתרון בשימוש של service מול factory הוא במעבר ל-es6, שם ניתן להמיר את הקוד של ה-service ולהשתמש במחלקות, לצורך המימוש של ה-service.
* **Material layout** בנוי מעל **css3 flexbox**. נועד כדי לתמוך ברספונסיביות. Layout יכול להיות row או column. Flex מקבל ערכים בקפיצה של 5 בין 0 ל 100 (מייצג 0% ועד 100%). אפשר להשתמש ב-breakpoint כמו למשל xs,sm,md,lg כדי לשנות את התצוגה בהתאם לגודל המסך, וזה קשור ל-mediaQuery ב-css.