תרגיל חפיפת Full Stack

מטרת התרגיל

בתרגיל זה נפתח web application. החל מבניית API, בניית frontend המתממשק איתו, ועד הכנת האפליקציה ל-deployment.

בתרגיל נעבוד עם:

- Node.js (עם TS/JS)
- React (עם TS/JS)
- Docker

דרישות מקדימות

- סביבת עבודה מועדפת (רצוי VSCode).
- Node.js
- Docker
- SQLite

דגשים למפתח

- יש לשמור על קונבנציות אחידות בקוד ולדאוג לקריאות שלו. ניתן להתייעץ עם החופף בעניין.
- מידור ואבטחה הן נקודות חשובות מאוד. משתמש (מכל סוג) אמור להיות רשאי לבצע אך ורק את הפעולות המותרות לו, ולא מעבר לכך. יש לוודא את הדברים האלה ברמת ה-API, ורצוי שגם ב-frontend מטעמי UX.
- המערכת אמורה להיות UXית, וצריכה להיראות ברמה סבירה. אין ציפייה לקבל מערכת שנראית מיליון דולר, אבל גם לא ברמת Hello World.
- ה-DB שמור בקובץ *chinook.db*, שנמצא בקובץ *chinook.zip* שהגיע עם התרגיל.
  הטבלאות הבאות אינן רלוונטיות לתרגיל: media_types, playlists, playlist_track, genres.
  בסוף התרגיל מצורף תרשים ERD המתאר את מבנה ה-DB כפי שהוא בקובץ.
  (הפורמט של ה-DB הוא sqlite)

על המערכת

המערכת אותה תפתחו מנהלת חנות וירטואלית לשירים.

לקוחות יכולים לחפש שירים ולרכוש אותם.

עובדים יכולים לראות את ההזמנות השונות, ולבצע פעולות על ההזמנות השונות.

מנהלים יכולים לבצע את מה שעובדים יכולים לעשות. בנוסף, למנהלים יש מסך סטטיסטיקות בו הם יכולים לראות נתונים רלוונטיים על תפקוד החנות.

let the fun begin

שלבים

1. חיפוש שירים

הפונקציונליות הראשונה שנפתח היא מנגנון חיפוש השירים.
לאחר שהלקוח נכנס למערכת (כפי שיתואר בשלב 2), תופיע לו תיבת חיפוש למציאת שירים.

התנהגות תיבת החיפוש:

- בעת הקלדה, הטקסט שהוקלד ישלח לשרת לעיבוד.
  הפניות לשרת צריכות להיות debounced.
- השרת יחזיר למשתמש את רשימת N השירים הרלוונטיים ביותר לחיפוש, עפ"י שם השיר, שם האלבום ושם הזמר.
- החיפוש יהיה אחוד. כלומר, המשתמש יקליד קלט אחד, ולא תהיה אפשרות לחפש לפי שם זמר/אלבום/שיר בלבד.
- בעת הגעת תוצאה מהשרת, רשימת התוצאות תעודכן.

רשימת התוצאות תכיל את:

- שם השיר
- שם האלבום
- שם הזמר
- מחיר השיר
- כפתור הוספה לסל

2. חיבור והרשמה

משתמשי המערכת (לקוחות ועובדים) צריכים להתחבר למערכת על מנת להשתמש בה, ויוכלו לצאת ע"י כפתור התנתקות.
במסך זה, המשתמש יוכל ליצור לעצמו חשבון ע"י הזנת הפרטים הבאים:

- שם משתמש
- סיסמה
- כל הפרטים הרלוונטיים בטבלת customers

לצורך שלב זה נוסיף טבלת משתמשים בשם users, שתכיל את השדות הבאים:

- שם משתמש
- **hash של הסיסמה שהוזנה** (להחליט על אלגוריתם hash בו יעשה שימוש)
- סוג משתמש (e = employee, c = customer)
- מזהה לקוח/עובד (מטבלת customers/employees בהתאם לסוג המשתמש)

(המשך השלב בעמוד הבא)

אין צורך לפתח אפשרות הרשמה לעובדים.
לצורך הבדיקה והעבודה על התרגיל, יש להוסיף ידנית נתונים לטבלת users ולחבר אותם לעובד הרלוונטי.

בזמן החיבור, המשתמש יזין את שם המשתמש והסיסמה. השרת לוודא שה-hash של הסיסמה שהוזנה זהה לזה ששמור ב-DB עבור השם משתמש שהוזן.
בעת חיבור מוצלח, המשתמש יקבל token חתום שמקנה לו את האפשרות להשתמש ב-API עם ההרשאות המתאימות.

3. סל קניות

לאחר שמשתמש מצא שיר שהוא רוצה לקנות - הוא מוסיף אותם לסל הקניות.
בעת לחיצה על כפתור ההוספה לסל, יש להוסיף את המוצר לרשימת המוצרים בסל של הלקוח (בטבלה ייעודית שתקימו ב-DB), על מנת שהסל יהיה זמין גם לאחר יציאה מהמערכת.

ניתן להגיע לסל הקניות מכפתור במסך הראשי (בו נמצא החיפוש).
מתוך סל הקניות יהיה אפשר למחוק שירים מהסל, ויהיה ניתן לבצע רכישה.

בעת רכישה, הלקוח יוכל לבחור את העובד שעזר לו (יש להתעלם משדה SupportRepId בטבלה customers, ולהוסיף אותו בטבלה invoices).
אין צורך לפתח מנגנון חיוב מסובך עבור רכישה. מבחינתכם מדובר בקסם שמטופל בצורה אחרת.

כפי שצוין בדגשים - מידור הוא דבר קריטי: משתמש יכול לראות רק את סל הקניות שלו, ואינו יכול לבצע פעולות על סלי קניות של משתמשים אחרים.

4. מסך מכירות

במסך המכירות, העובדים יוכלו לראות log של הרכישות שהתבצעו בחנות, עם הפרטים הרלוונטיים (לשיקולכם).

רק General Manager / Sales Manager או העובד שהוגדר בתור ה-support representative (לפי שדה SupportRepId) רשאים לראות את כל פרטי ההזמנה.
כל שאר העובדים רשאים לראות את פרטי ההזמנה, אך הפרטים יהיו מצונזרים, מלבד התו הראשון.
לדוגמה, עבור רכישה בה שם הרוכש הוא "David", עובד שאינו ה-rep יראה ששם הרוכש הוא "D***".

הרכישות יהיו ממויינות בסדר יורד, כך שהרכישה האחרונה תהיה למעלה, והרכישה הראשונה למטה.

העובדים יכולים לבטל רכישות ע"י מחיקתן. רכישות לא נמחקות מה-DB, אלה מסומנות כמחוקות, ואינן תקפות בשאר חלקי המערכת (כמו סעיף 5). מחיקה זו לא מסננת את הרכישה ממסך זה (הרכישה עדיין תופיעה כמחוקה אחרי ביצוע המחיקה).

5. מסך סטטיסטיקות

למסך זה יכולו לגשת רק אנשים בתפקיד "General Manager" או "Sales Manager".
במסך זה יופיעו 3 (או יותר) גרפים ו/או אינדיקציות המשקפות את תפקוד הסניף

יש לחשוב על נתונים רלוונטיים לעסק, ולקבל אישור על הרעיונות מהחופף לפני תחילת הפיתוח.

6. זמן לארוז את הכל

המערכת שלנו מוכנה! עכשיו רק צריך להרים אותה.
אנשי ה-IT התותחים של החנות מבינים עניין, ורוצים להרים את האפליקציה על GCP, ושתהיה scalable.

עליכם ליצור image של האפליקציה בעזרת docker, על מנת שהם יכולו להרים את החנות.
(לצורך הרמה של האפליקציה לוקאלית בקלות, מומלץ להשתמש ב-docker compose).

פיצ'רים נוספים

ניתן לבצע כל אחד מהתרגילים בנפרד, אין תלות בניהם.

1. חלק מהשליפות במערכת עלולות להיות כבדות ואיטיות ב-scale, ולרוב לחינם.
   הכניסו מנגנון cache לנתונים רלוונטיים במערכת.
   שימו לב שה-cache צריך להיות אחיד בין כל השרתים של המערכת, כדי לשמור על אחידות נתונים בצד המשתמש.
   ניתן להתייעץ עם החופף על הנקודות שרלוונטיות לשיפור ביצועים, ועל הטכנולוגיות בהן כדאי להשתמש.
2. ב-practice, שרת API לא אמור להגיש את ה-frontend למשתמש.
   במקום זה, הרימו nginx (בקונטיינר כמובן) שיגיש את הקבצים של ה-frontend למשתמש, ויהווה reverse proxy לשרת ה-API.
3. להוסיף 2-3 פיצ'רים נוספים שנראים לכם רלוונטיים במערכת. יש לאמת מול החופף לפני המימוש.

נספחים

ERD התחלתי