



8200

7160

יחידה

מרכז

שבועת השיבולים - תרגיל צה"ל



רקע

عقب גידול האוכלוסייה בבסיס "שבועת השיבולים", נוצר מצב בעיתוי מאוד במנוגרים של הבסיס. יש יותר מבקשי דיור מאשר דירות פנויות. הדבר גורם לבעה גדולה של ניהול החילימים הישנים במנוגרים, קושי ביצירת סדר והוגנות, וחוסר שקייפות בהשלכות השיבוץ. חיילים שגרים בקריות שמונה מוצאים את עצם חזרים כל יום הביתה עד שימצא להם פתרון.

מפקד הבסיס, אל"ם שיבולי, הורה על הקמת מערכת דיגיטלית שתאפשר ניהול מיטבי של הבעיה.

עליכם לבנות מערכת שתוכל:

- לנוהל בקלות את המנוגרים בבסיס
- לאפשר לייצור צדק באופן שבו משבצים חיילים למנוגרים

- לספק תמונה מלאה של תפוזת המגורים ומילוי נשאר בחוץ

הנחיות

1. עליכם לבנות מערכת (שרת API) לניהול מגורים חילילים בסיסי "שבוע השיבולים" בפייתון

המערכת תנהל:

- חילילים המבקשים מגורים בסיסי
- בתים מגורים - כרגע שניים (Dorm A, Dorm B), כל אחד מכיל 10 חדרים
- בכל חדר יש 8 מקומות לחילילים (סה"כ 80 חילילים לבניין מגורים, 160 לכל הבסיס)
- רשימת המתנה לחילילים שלא נמצא להם מקומות

2. שימוש בעזרים - מותר להשתמש בכל מה שלא AI (ולא AI-Mode)

3. אופן הבדיקה של התרגיל - הצוון מחושב לפי דגש על פונקציונליות עובדת, SOLID, OOP, קונבנציות פיתון לפי הסדר הזה

תוצר המערכת יהיה Rest API שבבסיסו הלוקוח יוכל לשלו CSV של חילילים, והשרת יבצע שיבוץ אוטומטי לפי כללי מוגדרים מראש. פירוט מופיע בהמשך לפי שלבים.

על הרכיבים (כל הפרטים על כל רכיב צריכים להופיע בקורס)

חיליל

חיליל הוא ישות בעל הנקודות הבאים:

- מספר אישי (מתחילה ב8, ומכיל מספרים בלבד)
- שם פרטי
- שם משפחה
- מין (זכר או נקבה)
- עיר מגורים
- מרחק מהבסיס בקילומטרים (מספר)
- סטטוס שיבוץ:
 - שובץ למגורים

- ממתין (לא שובץ)

בתים מגורים

בבסיס יש שני בתים מגורים בלבד:

- בית מגורים א'
- בית מגורים ב'

כמובן שכמו בחיקם האמיתיים, בעתיד יכולים להיבנות בתים מגורים נוספים, ולכך הקוד צרי
להיות פתוח להרחבות

כל בית מגורים מכיל:

- 10 חדרים
- בכל חדר 8 חילילים
- סה"כ 80 חילילים לבית מגורים, 160 חילילים בסיסי כולם

בסיס שבעת השיבולים

בסיס "שבעת השיבולים" הוא המקום שבו נמצאים בתים מגורים, ווחילילים. ניתן להניח שככל חיליל
שמגיע לבסיס באידיאל היה רוצה לקבל חדר שונה.

מערכת השליטה (API)

מערכת השליטה היא המשתק של המשתמש (איש לשירות, מש"קית ת"ש, קצינה קישור וכדומה)
מול המערכת.

באמצעות מערכת השליטה המשתמש יוכל:

- לייבא את רשימת החילילים מקובץ CSV (העלאת CSV)
- להריץ שיבוץ מחדש (לפי כללי קדימות)
- להציג דוחות מגורים
- להציג רשימת חילילים שלא שובצו
- ועוד דברים שיתווסף בהמשך

אופן הגשה

הגשת התרגיל תתבצע באמצעות GIT, ל-`repository` אישי המועד לתרגיל.

- הפיתוח יעשה ב-`Branch` בשם `dev`.
- בתוך `Branch` הראשי, `main` (או `master`), תהיה תמיד גרסה עדכנית **שמתקמפלת ועובדת**.
- לאחר סיום כל שלב בתרגיל, יש לעשות `Merge Request` מ-`dev` אל `master`.

שימוש לב:

- מומלץ לעשות הרבה **Commit**ים ל-`dev`, גם על שינויים קטנים.
- אין חובה שככל Commit ב-`dev` יהיה עובד - אבל מה שנכנס ל-`master` חייב להיות עובד.
- הקפידו על הודעות Commit ב-`dev` ברורות שמתארות מה השתנה.
- רק **האחרון commit ב-`master` הוא זה שיבדק**.

לקראת הגשה (עشر דקות אחרונות של התרגיל):

- יש לוודא בפעם الأخيرة כי `master` מכיל גרסא עדכנית **שמתקמפלת ועובדת**.
- יש לצרף `dump` של המידע ב-`db` שלכם (מדובר בקובץ אחד)
- במידה ויצרתם קבצי `csv` נוספים, צרפו אותם לפרויקט תחת התיקייה "`csv`"
- יש לוודא כי יש קובץ `README` עם פירוט מינימלי על הפרויקט
- יש למלא את [השאalon הגשה הסופי](#)

הערות

- חשבו על עיצוב המערכת בצורה סולידית (SOLID)
- השימוש בעקרונות OOP שלמדו:
 - הפרדה בין לוגיקת שיבוץ לבין ייצוג הנתונים
 - שימוש במשקים/מחלקות מופשטות במקום `switch`ים עוקיים
 - פתיחות להרחבה וסגירות לשינוי (Open/Closed Principle)

- חשבו איך המערכת תתמודד עם הרחבות עתידות:
 - הוספת בית מגורים שלישי
 - שינוי מדיניות עדיפות
 - הוספת תנאים מיוחדים (פרופיל, תפקיד, קבוע/סדיר וכו')
 - החלפה בין DBים מבלי לפגוע בLIBET הקוד
- התיחסות למקרי קצה - יש מקרים קצה, למשל: כמות השורות באקסל נמוך מכמות החדרים בפועל. עליהם לפטור כל מקרה קצה שיש לפי שיקולכם
- יש לבצע ולידציות בהתאם לצורך ולשיקול דעתכם - למשל, בהעלאת אקסל עם מספר אישי שמכיל אותן, אפשר לפסול את כל האקסל, או להחליט שורה אחת באקסל לא תעלה. אתם צריכים לפטור סוגיות כאלה בעצמכם, בהתאם למחשבה של "מה הדבר הנכון ביותר לפחות"
העקרונות של מדדי?
- אתם אלו שahlen מחליטים כיצד יראו המחלקות בקוד, ואיך הן מתקשרות אחת עם השנייה, ע"פ עקרונות SOLID ו-OOP
- שימושו לבشبוב א' וב' יקרה מצב שבו ריסוט האפליקציה (שיכולת לקרוות אחורי כל שמירה של קוד בגעל reload) יגרור מחיקה של המידע. אתם יכולים לפטור את זה בכל דרך שתבחרו
- תוכלו לבדוק את התוצרת שלכם באמצעות כלים שיודעים לבצע בקשות HTTP (כמו, Postman, cURL)
- השמות של routes יהיו בדיקוק כפי שהוגדרו בתרגיל. שאר השמות בתרגיל (מחלקות, משתנים וכו') יכולים לבחור על ידכם וצריכם לעמוד בקונבנציות של פיתוח

משימות

בשלבים א'-ב' המידע ישמר בזיכרון RAM בלבד (לא בקובץ, ולא בסיס נתונים, אלא - על ידי אובייקטים ומשתנים לפי המחלקות שהוגדרתם) בשלב ג' נעבור לשימירת המידע בסיס נתונים.

שלב א' (60%) - ממשו את מערכת השליטה
 צרו API המאפשר קריאת CSV ושיבוץ ראשוני בהינתן CSV של חיילים ([להלן דוגמה](#)), אתם צריכים לשבץ אותם למגורים.
 שימושו לב שיש חיילים שישארו בחוץ.

סדר העדיפות של השיבוץ (הנקרא גם "סטרטגיית השיבוץ") צריך להיות לפי קילומטר מהבסיס,
לדוגמה:

1. חיליל בעל מרחק של 100 קילומטרים מהבסיס ישובץ לפני חיליל בעל מרחק של 99 קילומטרים
2. במידה וישנו אותו מרחק, אפשר לבחור באופן שריורי

מימוש מפורט:

1. מימוש שרת

- צרו שרת בPython המסוג לקלט בקשות HTTP לפי הכלים שלמדו (למשל Flask)
- 2. הוספת Route מסוג Route POST - /assignWithCsv
 - הוסיפו לשרת שיצרתם Route POST מסוג POST שידע לקבל קובץ CSV
 - הערה: במידה ואתם משתמשים בשליחת הקובץ בבקשת עצמה, תוכלו להשתמש בקובץ יישירות בשרת
- 3. קריאת קובץ CSV

- ממשו לוגיקה בשרת שקוראת את הCSV וממיר אותה לאובייקטים הרלוונטיים
- צרו עבור כל שורה את האובייקטים המתאימים במערכת.
- 4. שיבוץ החילילים במקומות
 - התחילה לשבץ חילילים לפי סדר העדיפות לבתי המגורים ולהדרים עד שהבסיס מלא (160 מקומות).
 - אם אין מקום פנוי - החיליל נכנס לרשימת המתנה.
 - אתם מחליטים איך נראה השיבוץ בקורס

5. תשובה למשתמש - החזרו תשובה בפורמט JSON לשימוש המשתמש

- החזרו בבקשת HTTP את הנ吐ונים לפי הדרישות הבאות:
 - הדפסו סיכום:
 1. כמה חילילים שובצו בפועל
 2. כמה נשארו ברשימה המתנה
 - הציגו עבור כל חיליל:
 1. שובץ? כן/לא
 2. אם כן - בית מגורים + מספר חדר
 3. אם לא - ציון שהוא ברשימה המתנה
 - חשבו על המבנה JSON המתאים ביותר להחזרת הנ吐ונים המופיעים פה
- 6. בדקו את מה שעשיתם עם URL או Postman.

- דוגמה לבקשת URL שיכולה לעבוד:

```
curl -X POST http://localhost:5000/assignWithCsv \ -F "file=@soldiers.csv"
```

הערה - במידה וכבר בוצע **שיבוץ** בעבר ע"י המערכת, פניה **לroute** שוב תזרוס את מה שהיה בעבר, ותפעל רק **לפי המידע החדש**

נקודה למחשבה:

מדיניותות השיבוץ לבסיס יכולה להשנות (למשל, בעתיד יתווסף תנאי קדימות אחרים

כמו תפקיך, ותק, פרופיל, אוכלוסייה וכו').

נסו לבנות את המערכת כך שהלוגיקה של השיבוץ אינה "תפורה" ישירות למחלקה החיליל, אלא ניתנת להחלפה/שינוי בקלות.

שלב ב' (20%) – אל"ם שיבולי דורש נתונים

בשלב זה תוסיפו למערכת יכולות דוח ובקраה.

צרו Routeים נוספים בAPI:

1. **דוח תפוצה לפי בית מגורים - /space**

- מימוש Route מסוג GET המחזיר עבור כל בית:

■ מספר החדרים המלאים

■ מספר החדרים החלקיים

■ מספר החדרים הריקים

2. **דוח רשימת המתנה - /waitingList**

- מימוש Route מסוג GET המחזיר את כל החילילים שלא שוכנו, לפי סדר עדיפות השיבוץ.

3. **חיפוש חיליל לפי מספר אישי /search**

- מימוש Route מסוג GET המחזיר פרטים על החיליל לפי מספר אישי:

■ שוכן?

■ אם כן – איפה (בית+חדר)

■ אם לא – ברשימה המתנה.

על התשובות לחזור כJSON (שאתם מחייבים איך נראה). כמו כן, אתם מחייבים כיצד לקבל את הפרמטרים בבקשת HTTP ועל הסטטוס חזרה

שלב ג' (20%) - התממשקות עם בסיס נתונים (DB)

רקע:
מה הבעיה במא שעשיתם עד עכשו? כסוגרים את התוכנית (או שהמחשב נכבה), הלא כל המידע במערכת!

מטרה:

להעביר את הנתונים במערכת (חילילים, בתים מגורים, חדרים, שיבוצים) לאחסון מתמשך בבסיס נתונים SQLite - לפי מה שנלמד בקורס.

משימות:

1. הקמת DB

- הקימו DB בעזרת בו תשימושו בשביל לאחסן את המידע של המערכת

2. אתחול סכמה ב-DB - initializeScheme

- הוסיפו לשרת יצירתם Post Route שיאתחול את הסכמה ב-DB. אין צורך בשלב זה לשים מידע ב-DB.
- **שים לב!** אתם מחייבים כיצד הסכמה תיראה, כמוון שאפשר לבחור ליצור כמה טבלאות

3. עדכון ה-**API** לעובד עם DB

- כל Route משלב אי ו-ב' צריך לעבוד מול בסיס הנתונים:
 - קריית CSV בשלב אי תכenis חילילים לטבלאות הרלוונטיות.
 - לוגיקת השיבור תעדכן את טבלאות השיבור/סטטוס.
 - דוחות (space, waitingList, Chiil) יופקו על בסיס שאילתות DB.

טיפ: חשבו על שימור ה-**EEP** וההפרזה הלוגית:

- וודאו שהמחלקות שתכתבם אכן קיימות כאובייקטים לוגיים.
 - הגישה לבסיס הנתונים לא "נולגת" לכל מקום בקוד, אלא מרכזת בשכבה מתאימה.
- שימוש לבסיסו של זה סגירת התוכנית ופתיחה מחדש לא אמורה להשפיע על זמינות הנתונים, מעכשו המערכת שרידה!

שלב ד' (20%) - בונוס (מי שסיים את א'-ג')

💡 שימוש ♥ – עדייף לסייע עד שלב ג' באופן מלא ובינוי טוב לפיה העקרונות שנלמדו, מאשר לרווח להלאה. במידה ושלב הבונוס פוגע בתוצר של שלבים א'-ג' אין להגיש אותו!

הרchieבו את המערכת כך שתתמודד טוב יותר עם שינויים לארוך זמן:

1. עזיבת חילילים - /release

- הוסיף Route מסוג POST ל"שחרור" חיליל מחדר לפי מספר אישי
- לאחר שחרור מקום:
- המערכת צריכה לשבץ אוטומטית את החיליל הבא בתור מרשימת המנתנה. ע"פ האסטרטגייה שיבוץ שבה האקסל עלה
- עדכנו גם את בסיס הנתונים בהתאם (מחיקת שיבוץ / עדכון סטטוס).

2. הוספת אסטרטגיית שיבוץ חדשה - PazamStrategy

- הוסיף לכל חיליל שדה "דרגה". הדרגה תהיה 5-1 כאשר 5 זה הרמטכ"ל ו-1 זה טוראי
- בנו אסטרטגיה חדשה לפיה במידה ויש חילילים עם אותו מרחק מהבסיס, העדיפות תקבע לפי הדרגה.
- בבקשת - POST /assignWithCsv אותה מימוש בעבר, הוסיף אפשרות לבחור באסטרטגיה הבסיסית (מיון לפי מרחק בלבד) או החדשה (מיון לפי מרחק, ואם המרחק זהה אז דרגה). אין לפגוע בבקשת POST שכבר עובdot. חשבו איך לעשות את זה בצורה OCP

3. הוסיף Post Route חדש של הוספה במקום דרישת appendWithCsv

- הוסיף לשרת שיצרתם Route חדש מסוג POST שידע לקבל קובץ CSV
- כל חיליל חדש יתווסף למערכת. במידה וקיים לו חדר פנוי הוא ישובץ, במידה ולא, יצטרף לרשימה המנתנה. במידה ויש ניסיון להעלות מספר אישי שכבר קיים, טיפולו במקרה שיקולכם