



8200

יחידה

7160

מרכז

שבועת השיבולים - תרגיל צה"ל



פרק

עקב גידול האוכלוסייה בבסיס "שבועת השיבולים", נוצר מצב בעייתי מאוד במגורים של הבסיס. יש יותר מבקשי דיור מאשר דירות פנויות. הדבר גורם לבעה גדולה של ניהול החיללים הישנים במגורים, קושי ביצירת סדר והונgot, וחוסר שקיפות בהחלהות השביב. חיילים שגרים בקריית שמונה מוצאים את עצם חזריהם כל יום הביתה עד שימצא להם פתרון.

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

עליכם לבנות מערכת שתוכל:

- **לנהל בקלות את המגורים בבסיס**

- לאפשר ליצרן צדק באופן שבו משבציהם חילילים למגורים
 - לספק תמונות מצלב מלאה של תפוזת המגורים וממי נשאר בחוץ
-

הנחיות

1. עלייכם לבנות מערכת (שרת API) לניהול מגורי חילילים בסיסי "שבוע השיבולים" בפייתון המערכת תנהל:

- חילילים המבקשים למגורים בסיסי
 - בתים מגוריים - כרגע שניים (Dorm A, Dorm B), כל אחד מכיל 10 חדרים
 - בכל חדר יש 8 מקומות לחילילים (סה"כ 80 חילילים לבניין מגוריים, 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 - הציון מחושב לפי דגש על פונקציונליות עובדת, OOP, SOLID,
- קונבנציות פיתון לפי הסדר הזה

творך המערכת יהיה Rest API שבבסיסו הלוקח יכול לשЛОוח קובץ 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 בקוד)

חיליל

חיליל הוא ישות בעלת הנתונים הבאים:

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

- שובץ למגורים
 - ממתין (לא שובץ)
-

בתיהם

בבסיס יש שני בתים למגורים בלבד:

- בית למגורים א'
- בית למגורים ב'

כਮון שכמו בחיקם האמתיים, בעתיד יאפשרו להיבנות בתים למגורים נוספים, ולכן הקוד צריך להיות פתוח להרחבות

כל בית למגורים מכיל:

- 10 חדרים
- בכל חדר 8 חילילים
- סה"כ 80 חילילים לבית למגורים, 160 חילילים בסיס כולל

בסיס שבעת השיבולים

בסיס "שבעת השיבולים" הוא המקום שבו נמצאים בתים למגורים, והחילילים. ניתן להניח שככל חיליל שmagiu לבסיס באידיאל היה רוצה לקבל חדר שונה בו.

מערכת השליטה (API)

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

באמצעות מערכת השליטה המשתמש יוכל:

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

אופן הגשה

הגשת התרגיל תבוצע באמצעות GIT, ל-repository אישי המוצע לתרגיל.

- הפיתוח ייעשה ב-Branch בשם `dev`
- בתוך Branch הראשי, (או `main`), 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 ברורות שמתארות מה השתנה.
- רק **האחרון ב-Master הוא זה שיבדק**

לקראת הגשה (עشر דקotas אחרונות של התרגיל):

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

הערות

- חשבו על עיצוב המערכת בצורה סולידיית (SOLID)
- השתמשו בעקרונות OOP שלמדתם:
 - הפרדה בין לוגיקת שיבוץ לבין ייצוג הנתונים
 - שימוש במשקדים/מחלקות מופשטות במקום `switch`ים ענקיים
 - פתיחות להרחבה וסגירות לשינוי (`Open/Closed Principle`)
- חשבו איך המערכת תתמודד עם הרחבות עתידיות:

- הוספת בית מגורים שלישי
 - שינוי מדיניות עדיפות
 - הוספת תנאים מיוחדים (פרופיל, תפקיד, קבוע/סדיר וכו')
 - החלפה בין DBים מבליפגיעה בLIBT הקיים
 - התייחסות למקרי קצה - יש מקרי קצה, למשל: כמות השורות באקסל נמוך מכמות החדרים בפועל. עליכם לפטור כל מקירה קצה שיש לפי שיקולכם
 - יש לבצע וליזציג בהתאם לצורך **ולשיקול דעתכם** - למשל, בהעלאת אקסל עם מספר אישי שמכיל אותן, אפשר לפסול את כל האקסל, או להחליט שורה אחת באקסל לא תעללה. **אתם נדרשם לפטור סוגיות כאלה בעצמכם, בהתאם למחשבה של "מה הדבר הנכון ביותר לפי העקרונות של מدت?!"**
 - אתם אלו שמחיליטים כיצד יראו המחלקות בקוד, ואין הן מתקשרות אחת עם השנייה, ע"פ עקרונות SOLID ו-OOP
 - שימושו לבشبלב א' וב' יקרה מצב שבו ריסוט האפליקציה (שיכולת לקרוות אחורי כל שמירה של קובץ בגלל hot-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 `outputs` יהיו בדיקוק כפי שהוגדרו בתרגיל. שאר השמות בתרגיל (מחלקות, משתנים וכו') יכולים להיבחר על ידכם וצריכים לעמוד בקונבנציות של פיתוח
-

משימות

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

שלב א' (60%) – ממשו את מערכת השליטה

צרו API המאפשר קריאת CSV ושיבוץ ראשוני

בהתנתן CSV של חייבים (להלן דוגמה), אתם צריכים לשבץ אותן למגורים.

שימוש לב שישחייבים שישארו בחוץ.

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

1. חייל בעל מרחק של 100 קילומטרים מהבסיס ישוב לפני חייל בעל מרחק של 99 קילומטרים

2. במידה וישנו אותו מרחק, אפשר לבחור באופן שרירותי

מימוש מפורט:

1. מימוש שירות

- צרו שירות בPython המסוג לקלוט בקשות HTTP לפי הכלים שלמדתם (למשל בFlask)

2. הוספת Route מסוג POST - /assignWithCsv

- הוסיף לשרת شيئاוות Route POST מסוג שידע לקבל קובץ CSV

הערה: במידה ואתם משתמשים בשילוח הקובץ בבקשת עצמה, תוכלו להשתמש

בקובץ יישירות בשרת

3. קריאת קובץ CSV

- ממשו לוגיקה בשרת שקוראת את CSV וממיר אותה לאובייקטים הרלוונטיים

- צרו עבור כל שורה את האובייקטים המתאימים במערכת.

4. שיבוץ חילילים במגורים

- התחילו 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 בקובץ

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 cURL או Postman:

- דוגמה לבקשת cURL שיכולה לעבוד:

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

הערה - במידה וכבר בוצע שיבוץ בעבר ע"י המערכת, פניה `leoute` שוב תדרוס את מה שהיה בעבר, ותפעל רקippi המידע החדש

נקודה למחשבה:

mdiיניות השיבוץ לבסיס יכולה להשתנות (למשל, בעתיד יתווסף תנאי קדימות אחרים כמו תפקיד, ותק, פרופיל, אוכלוסייה וכו').

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

שלב ב' (20%) - אל"ם שיבולי דורש נתוניים

בשלב זה תוסיפו למערכת יכולות דו"ח ובקרה.

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 הഫרטרים בבקשת HTTP ועל הסטטוס חוזר

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

רקע:

מה הבעיה במה שעשיתם עד עכשיו? כסטודנטים את התוכנית (או שהמחשב נכבה), הלא כל המידע במערכת!

מטרה:

להעביר את הנתונים במערכת (חייבים, בתים, מגורים, חדרים, שיכוצים) לאחסן מותאם בסיס נתונים SQLite - לפי מה שנלמד בקורס.

משימות:

1. הקמת DB

- הקימו DB בעזרת בו תשימושו בשיל לארח את המידע של המערכת

2. אתחול סכמה ב-DB /initializeScheme

- הוסיפו לשרת שיצרתם מסוג 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` לעבוד עם DB

- כל Route משלב אי ו-בי צריך לעבוד מול בסיס הנתונים:
 - קריית CSV בשלב אי תכניס חיילים לטבלאות הרלוונטיות.
 - לוגיקת השיבוץ تعدכן את טבלאות השיבוץ/סטטוס.
 - דוחות (space, waitingList, חיוף חייל) יופקו על בסיס שאלות DB.

טיפ: חשבו על שימור ה-`EEP` וה הפרדה הלוגית:

- וודאו שהמחלקות שתכתבם עדיין קיימות כאובייקטים לוגיים.
- הגישה לבסיס הנתונים לא "נעלגת" לכל מקום בקוד, אלא מרוכזת בשכבה מתאימה.
**שים לב שבסוף שלב זה סגירת התוכנית ופתיחה מחדש לא אמורה להשפיע על זמינות הנתונים!
מעכשו המערכת שרידה!**

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

 **שים ♥ – עדיף לסייע עד שלב ג' באופן מלא ובנוי טוב לפי העקרונות שנלמדו, מאשר לרוץ הלאה. במידה ושלב הבונוס פוגע בתוצר של שלבים א'-ג' אין להגיש אותו!**

הרchievio את המערכת כך שתתמודד טוב יותר עם שינויים לאורך זמן:

1. עזיבת חיילים - `/release`

- הוסיפו Route מסווג POST ל"שחרור" חייל מחרדר לפי מספר אישי

- לאחר שחרור מקום:

- המרכיבת צריכה לשבץ **אוטומטיבית** את החיליל הבא בתורו מרישימת ההמתנה. ע"פ האסטרטגיית שיבוץ שבה האקסל עליה
- עדכנו גם את בסיס הנתונים בהתאם (מחיקת שיבוץ / עדכון סטטוס).

2. הוספת אסטרטגיית שיבוץ חדשה - PazamStrategy

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

3. הוסיפו Post Route חדש של הוספה במקום דרישת /appendWithCsv

- הוסיפו לשרת שיצרתם Route חדש מסוג POST שיידע לקבל קובץ CSV
- כל חיליל חדש יתווסף למערכת. במידה וקיים לו חדר פנוי הוא ישובץ, במידה ולא, יctrarף לרישימת ההמתנה. במידה ויש ניסיון להעלות מספר אישים שכבר קיימים, טיפולו במקרה שיקולכם