שלום  
  
מצרף הסבר קצר לפיתרון ולשיקולים שהנחו אותי בבחירת הארכיטקטורה

הנחות מוצריות

1. יוזר יכול להגדיר התראה שמבוססת על מקור הטיסה יעד הטיסה תאריך הטיסה ומחיר
2. היוזר יוכל להגדיר התראה גם על טיסה לכיוון אחד וגם על טיסת הלוך חזור
3. כל עוד היוזר לא מבטל את ההתראה הוא ימשיך לקבל התראות גם אם כבר קיבל התראות, למשל, אם יוזר הגדיר התראה כאשר המחיר יורד מ- 500 וחברה מסויימת מפרסמת שהמחיר הוא 400 ואחריה חברה אחרת מפרסמת שהמחיר הוא 450 – היוזר יקבל את שתי ההתראות (אפרט גם אפשרות לשנות הנחה זאת במידת הצורך)

בהסתמך על ההנחות האלה

1. להתראה **אין** PK אלא היא מבוססת על מספר פרמטרים שאינם UNIQUE – לכן בחרתי לשמור את ההתראות ב- DB שמאפשר אינדקס על **מספר** שדות (למשל SQL Server, Oracle, MongoDB וכו') והגדרתי אינדקס על אותם שדות לצורך שליפה מהירה
2. מכיוון שלכל טיסה יש מקור ויעד (גם טיסות לכיוון אחד וגם טיסות הלוך-חזור) ומכיוון שהשדות האלה הם הראשונים שנדרשים לשליפת ההתראות מה- DB בחרתי להגדיר שדה מיוחד שקראתי לו FlightAbbreviation  
     
   שדה זה הוא בן 6 אותיות שהם הרכב של מקור ויעד הטיסה למשל PARMAD מייצג טיסה מפריס למדריד ושדה זה ישמש **כאינדקס הראשון** של טבלת ההתראות  
   מכיוון שאינדקס הוא עץ שהפעלתו נקבעת לפי סדר השדות הדבר יאפשר לנו להגדיר שדה אחד במקום שניים וישפר את יעילות השליפות ועל הדרך גם נחסוך במקום  
   שדה זה יאפשר גם לבצע Partition על ה- DB מה שיאפשר שליפות עוד יותר מהירות

פירוט קצר של הארכיטקטורה

1. נבנה אוסף של Services כאשר כל אחד פועל מול ספק כלשהו, במידה והספק 'שופך' את המידע שלו לתור או שולח webhook ה- Service הרלוונטי יאזין להם, במידה והספק מספק רק API ה- Service הרלוונטי יפנה אליו עם Timestamp כדי לקבל את **השינויים** בלבד
2. כל אחד מה- Services הנ"ל 'יתרגם' את המידע שמתקבל מהספק לאובייקט **אחיד** שנקרא Flight (מפורט בתרשים) ויכתוב אותו לתוך Flight PubSub שניתן לממש בכל סוג שהוא (Kafka,rabbitmq,servicebus)
3. ההודעות יקראו מה- PubSub ע"י ה- AlertMatcher שיבדוק האם קיימות התראות רלוונטיות לכל הודעה, מכיוון שהמידע של ההתראות מאונדקס ב- DB הדבר יאפשר שליפות מהירות
4. במידה וימצאו התראות רלוונטיות AlertMatcher יוסיף לכל אחת מהן את המידע הנוסף שרלוונטי ליוזר שהוא חברת התעופה והמחיר כך שיווצר אובייקט שיקרא AlertToSend ואת האובייקט הזה הוא ישפוך כהודעה ל- AlertsToSend PubSub
5. ל- PubSub הזה יאזין AlertSender שישלוף מטבלת UserDevice את פרטי ה- Device של היוזר ויעביר את ההודעה ל- Service שיבצע את שליחת ה- Push notification   
     
   טבלת UserDevice מתבססת על PK ולכן אפשר לשמור אותה גם ב- DB שתומך באינדקסים (כפי שציינתי למעלה) כאשר UserName הוא PK וגם במבני נתונים שמתבססים על PK בלבד כמו Redis

מענה לנושא של הפסקת התראות

1. במידה ויוגדר שיוזר **לא** צריך לקבל התראות יותר יקרות לאחר שכבר קיבל התראה ניתן להוסיף לאובייקט Alert את השדה ‘LastAlertPrice’   
   (הוספתי אותו בהערה בקוד של ה- CRUD API)
2. שדה זה יקבל את המחיר של ההתראה שנשלחה ליוזר ובשליפה נתייחס אליו, כך כל טיסה חדשה שתהיה יקרה יותר מהטיסה שהיוזר כבר עודכן עליה לא תשלח  
     
   למשל אם היוזר ביקש לקבל התראה כאשר המחיר יורד מ- 500 והתקבלה טיסה שעולה 400, נעדכן את השדה ‘LastAlertPrice’ בערך 400 ואז כאשר תגיע טיסה שמחירה 450 לא נשלח את ההתראה הזאת

זהו, עד כאן, מקווה שלא חפרתי יותר מדי, היה חשוב לי להסביר את הפתרון



[This Photo](https://en.wikipedia.org/wiki/Smiley) by Unknown Author is licensed under [CC BY-SA](https://creativecommons.org/licenses/by-sa/3.0/)