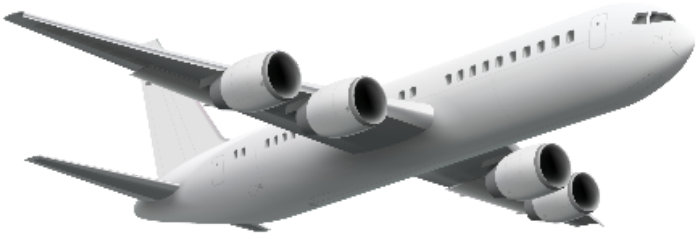
פרויקט מערכת טיסות

קורס פייתון - John Bryce

חלק ג' - Web API



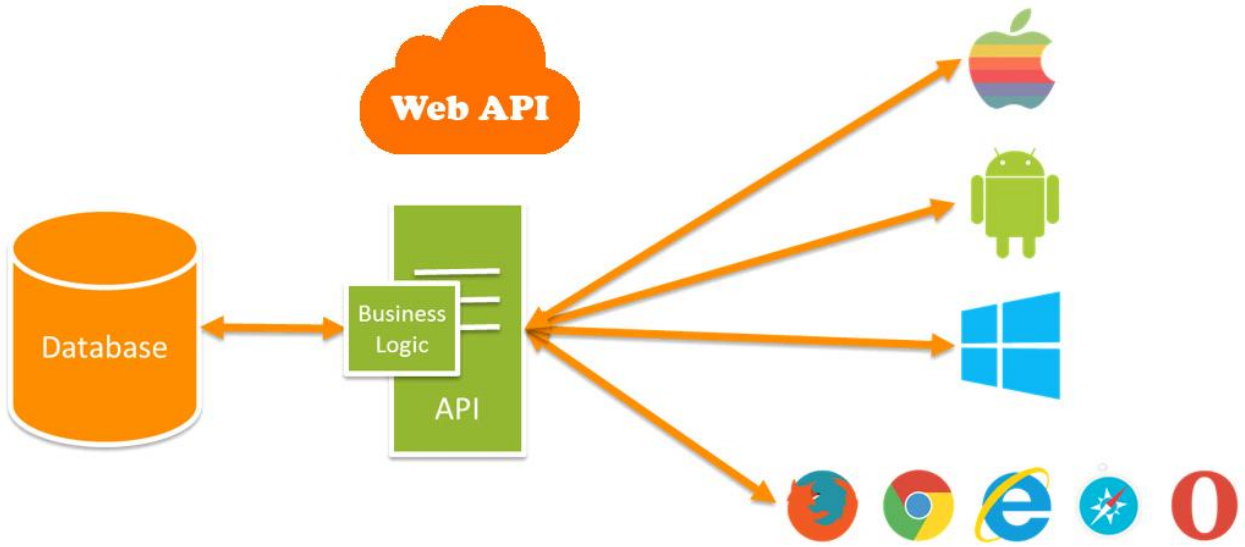
JOHN BRYCE
תלמד ה'-טק. זה עובד!
a matrix company

תיאור:

בשלב זה של הפרויקט, האפליקציה שלנו כבר יודעת לספק את כל הלוגיקה העסקית, יש לה Database וגם מחולל נתונים. אבל, היא עדיין מבודדת מן העולם החיצון, ולא ניתן לחבר אותה ל-Clients.

לשם כך אנחנו נזדקק ל-**Web API** (או במילים אחרות - ממשק אינטרנטי) אשר יחשוף החוצה את כל הפעולות והפונקציונאליות של המערכת שלנו.

את ה-Web API אנחנו נבנה באמצעות **Flask** בו למדנו להשתמש בשיעורים האחרונים. מטרתו היא לספק את הפעולות הסטנדרטיות של REST, והן: GET, PUT, POST, DELETE ו- PATCH.



אותנטיקציה (Authentication):

לצורך ה-authentication נשתמש בטכנולוגיית JWT כפי שלמדנו.

1. כאשר המשתמש יבצע login אנו נבצע פעולת decode ונחזיר אליו jwt token.
2. כאשר המשתמש יבצע signup המערכת תייצר עבורו public id ותשלח לו.
3. אנו נממש decorator ונעטוף כל מתודה (מלבד המתודות של Anonymous), כדי לאכוף למי מותר להשתמש בה.

את ה-secret key של ה-jwt יש לאחסן בקובץ config.

Rest Controllers:

אנו נייצר ארבעה Controllers, אחד לכל ישות, לצורך גישה למערכת:

- Anonymous Controller
- Customer Controller
- Airline Controller
- Admin Controller

ב-Controllers אנחנו נגשר (באמצעות Rabbit) אל כל הפונקציות שב-Facades שלנו.

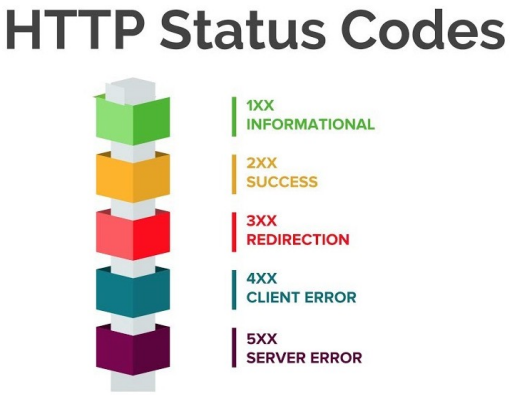
Routes and Query Parameters:

לכל פונקציה ניתן Route ייחודי משלה. נקפיד לתת שמות משמעותיים.
בפונקציות מסוג GET ניתן להשתמש ב-query parameters.

Status Codes:

על השרת להחזיר את הסטטוסים הבאים:

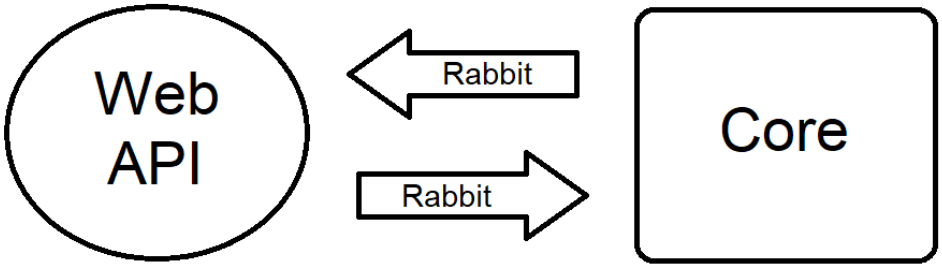
- 200 - כאשר הפעולה הצליחה (נשתמש בזה בפעולות GET או POST)
- 204 - כאשר הפעולה הצליחה ואין תוכן שצריך להחזיר (נשתמש בזה ב-PUT ו-DELETE)
- 400 - כאשר משהו אינו תקין בבקשה שהגיעה
- 401 - כאשר למשתמש אין גישה למערכת (לא סיפק token, למשל)
- 403 - כאשר המשתמש מנסה לעשות פעולה לא מורשית (למשל customer שמנסה לבצע פעולה של airline)
- 500 - כאשר התרחשה שגיאה כלשהי בצד השרת



בניית תורי Rabbit:

ההתממשקות בין ה-Web API למערכת שלנו (ה-Core) תבוצע באמצעות שני תורים של Rabbit.

1. התור הראשון, מה-Core ל-Web API, יטפל במידע החוזר מן המערכת, או בשגיאות כאלה ואחרות.
2. התור השני, מה-Web API ל-Core, יטפל בשליחת הבקשות למערכת.



עבודה אסינכרונית (Async):

- כל פעולות הקריאה מן ה-DB ב-Core יהיו אסינכרוניות.
- לעומת זאת, פעולות הכתיבה ל-DB יהיו סינכרוניות.

כיצד המערכת לא תבלבל בין ההודעות ב-Rabbit ?

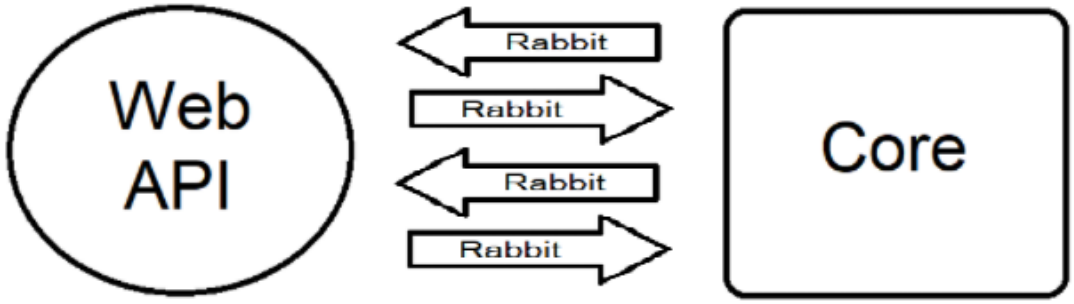
את בעיה זו נפתור באמצעות Correlation ID.
לכל בקשה תהיה מספר מזהה זה, שבאמצעותו כל Thread ידע מה הבקשה בה הוא מטפל.

ה-Consumer יוריד מהתור רק את הבקשות התואמות לו (לפי ה-Correlation ID)

אתגר (לא חובה): שיפור המערכת על ידי הוספת תורים:

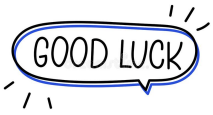
על מנת לאפשר ביצועים מערכת מהירים יותר (במיוחד כאשר מגיעות מספר רב של בקשות),
נדאג שכל הפונקציות ב-Controllers יהיו אסינכרוניות (בנוסף לאלו שב-Core)

כמו כן, נוסיף עוד שני תורי Rabbit (אחד לכל כיוון):



דגשים:

* לא לשכוח לעדכן את קובץ ה-requirements (או pepenv).
* יש להשתמש ב-logger כדי לתעד את הפעולות השונות.
* כתוב קוד מסודר, קריא וברור. הוסף הערות במידת הצורך.
* יש להעלות את הקוד ל- GIT בסיום.



בהצלחה!