תוספות לפרוייקט

1. הוספת LOGGER:
הוסף logger framework לפרוייקט לפי בחירתך (מומלץ בחום log4net)
בחר flow בפרוייקט משלב ה- web api ועד לשלב ה DAO (כולל האותנטיקציה וכולל ה- facade) ושם תעד בלוג כל כניסה ויציאה לפונקציה. כל פרמטר שנכנס וערך מוחזר. תעד כל שלב לוגי שקורה. כמובן **<u>שכל</u>** exception יש לתעד.
השתמש בכמה רמות לוג שונות המתאימות לכל הערה (לדוגמא exception יתועד ברמת לוג של error וכו')
* ***בונוס:*** כתוב אפליקצית #C אשר מקבלת כארגומנט קובץ קוד של סי שארפ ומחפשת try catch בתוכו. האפליקציה תדרוס את הקובץ ותשנה אותו ע"י הוספת קוד הכותב ללוג.
אם הצלחת, הרץ את האפליקציה על הפרוייקט וכך יהיה תיעוד בלוג לכל אחד מה exceptions

2. הוספת CACHE:
הוסף זיכרון מטמון לפרוייקט לפי בחירתך (מומלץ בחום Redis Cache)
בעת טעינת מערכת טען ל- CACHE את רשימת השאילתות הקשורות למידע שלא סביר שישתנה, לדוגמא שמות של מדינות.
בנה מנגנון שבכל פעם שמידע סטטי כגון "רשימת המדינות" ינסה להיקרא, בדוק תחילה אם המידע קיים כבר ב CACHE, אם כן – קרא אותו מה CACHE (ולא מהדאטא בייס). אם לא, קרא אותו מהדאטא בייס וכתוב אותו ב- CACHE (לחסוף זמן בפעם הבאה)
* ***בונוס:*** כעת השתמש באותו המנגנון לשאילתות המביאות מידע מבסיס הנתונים. לדוגמא רשימת טיסות- אם המידע קיים ב- CACHE קרא אותו משם אחרת הרץ שאילתא וכו'.
שים לב: אם תרוץ שאילתא כלשהי המשנה את המידע (כגון הוספת טיסה, שינוי או מחיקה) אז מחק את המידע השמור ב- CACHE (בכדי לגרום לו להיטען מחדש בפעם הבאה)

3. הוספת Rabbit MQ:
הוסף שירות messaging queue (מומלץ בחום Rabbit MQ)
בחר פונקציה ב- web api ובימקום "לטפל" בבקשה מיד, שים את הבקשה ב queue. והמתן לתשובה ב queue נפרד.
ב task אחר האזן לבקשות ובכל פעם שמגיעה בקשה- טפל בה ואכסן את התשובה ב- queue המיועד לתשובות. כמובן שאת התשובה יחזיר ה- thread מה web api
* ***בונוס:*** העבר את כל מתודות ה web api לעבוד דרך ה queue
** ***בונוס:*** בנה שירות window service אשר מאזין לבקשות ומטפל בהם בתהליך נפרד

בהצלחה