

כל השליפות ממסד נתונים, מייבא את כל המידע הרלוונטי והדרוש בשאילתה אחת,

לדוגמה ברגע שלקוח מתחבר במקום להביא רק את כל מסעדות בסניף בית שלו, ובכל פעם שמחליף סניף לגשת שוב לdatabase ולהביא את הסניף הרלוונטי, אנחנו מביאים ישירות את כל המסעדות בסניפים השונים ובכל שינוי סניף רק מעדכנים את התצוגה.

דוגמה נוספת היא ביצוע רכישה, בעת קנייה ושמירת מוצרים בעגלת הקניות, המערכת לא פונה למסד בכדי לשמור נתונים, אלא שומרת את המוצרים שכרגע בעגלה במפה ומבצעת פנייה ושמירה במסד רק ברגע בו בוצעה הקנייה, מהלך שכזה חוסך פניות מיותרות למסד ברגע שמשתמש מוחק מוצרים המעגלה או מבטל את כל ההזמנה.

\*סניף – מחוז – צפון,דרום,מרכז



QueryControl מרכזת גישה למחלקות שונות הקשורות לשאילתות, כגון UserQuery, OrderQuery ו-ManagersQuery. בכל מחלקה כזו מחזיקה את השאילתות הרלוונטיות לתחום שלו, ובכך אנו יודעים לאיזה מחלקה לגשת בכדי להוסיף\לשנות שאילתות וגם עוזרת במניעת שכפול קוד. על ידי שימוש במופעים סטטיים, הקוד מונע יצירת מופעים חדשים בכל פעם שיש צורך בשאילתה, זה משפר את הביצועים והופך את הקוד ליעיל יותר ומבטיח שמחלקות שאילתות מנוהלות באופן עקבי.

מחלקת- ClientMainController כל מתודה מטפלת בסוג מסוים של בקשה, עיצוב מודולרי זה מאפשר להסתגל בקלות לשינויים. אם יש צורך להוסיף בקשה מסוג חדש או אם יש צורך לשנות בקשה קיימת, בעזרת ClientRequest Enum קל לזהות את הבקשה הנדרשת, ואפשר לעבור ישירות לcontoller שמבקש את הבקשה ולבצע שינויים רלוונטים.

ScreenLoader לטעינת מסכים שונים בהתבסס על ה-Screen enum. עיצוב זה גמיש מכיוון שאפשר בקלות להוסיף מסכים חדשים או לשנות את המסכים הקיימים על ידי הוספה או שינוי enum של מסך.



במחלקה CheckoutScreenController יש מתודה בשם setupRadioButtons שכרגע מאתחלת ו"משבית" את RobotRadioBtn שזו כפתור בחירת משלוח באמצעות רובוט. למרות שה-RobotRadioBtn מושבת כרגע, הלוגיקה של הכפתור קיימת ומיושמת בקוד. בגרסאות עתידיות של האפליקציה נוכל לאפשר בקלות את RobotRadioBtn ולהפעיל את הפונקציונליות הקשורה בו מבלי להזדקק לשינויים משמעותיים בבסיס הקוד הקיים.

* כל שדרוש הוא למחוק את שורת ההשבתה של הכפתור