**תמלול**

**[שקופית 1]**

היי לכולם, אני {שם פרטי}, מפתח בצוות {שם צוות}.

בחצי השעה הקרובה אנחנו נדבר על הספרייה Entity Framework Core 6 ונראה כיצד משתמשים בה בפרויקט קטן שהכנתי.

רגע לפני שנתחיל, בגלל שאני לא מולכם אז כולם לבטל השתק, ואני חייב להראות לכם 3 בדיחות על NHibernate ש-ChatGPT סיפרה לי (כן זאת אישה, ולמי שלא מכיר, זו בינה מלאכותית שאפשר להתכתב איתה).

**[שקופית 2 – ללא דיבור]**

**[שקופית 3]**

אוקי, יש לנו חצי שעה, אז אני מתחיל.  
על מה אדבר?

אני אתחיל בלספר בקצרה על הספרייה ומתי לא להשתמש בה.  
אענה על השאלה "כיצד הספרייה מכירה את הטבלאות מבסיס הנתונים?".

לאחר מכן, אתייחס לשאלה "מה עושים כאשר כבר קיים בסיס נתונים?" ועל השאלה "מה עושים כאשר אין?".

ואז, אציג פרויקט עובד עם Swagger.

ובסוף, מקווה שיהיה לנו קצת זמן לענות על שאלות.

**[שקופית 4]**

רגע לפני שנדבר על EF Core, אני רוצה לעשות סוג של יישור קו ולציין מספר שינויים שהגיעו עם Dot Net Core.  
Dot Net Core מגיע עם מגוון רחב של שיפורים ויכולות, ביניהם שיפור משמעותי בביצועים, Cross Platform, יכולות Injection (הזרקה) והקבצים web.config ו-global.asax לא קיימים יותר - במקומם יש appsettings.json ו-Program.cs.  
כמובן, יש עוד הרבה מה להרחיב אבל זה לא רלוונטי לפרזנטציה זו, אז נשאיר את זה לפרזנטציה אחרת.

**[שקופית 5]**

אז מה זה בכלל EF Core? אז, זו ספריה שמטרתה לספק לנו דרך יעילה ונוחה לגשת לבסיס הנתונים שלנו ולבצע מולו שאילתות.

אני מאמין שלחלק ממכם יצא לעבוד עם NHibernate ולפעמים נאלץ או העדיף לכתוב שאילתת SQL ידנית, כי NHibernate לא תמיד מנגן כמו שהיינו רוצים.  
אז ב-EF Core הגישה לבסיסי נתונים נעשית באמת ללא צורך בכתיבת שאילתות SQL. כל פעולת CRUD מבוצעת בעיקר בעזרת סינטקס LINQ ובצורה מאוד אינטואיטיבית. כאשר, יש תמיכה במגוון סוגי בסיסי נתונים והקוד שנצטרך לכתוב יהיה זהה בכולם.

אנחנו תכף נראה בפרויקט שהכנתי.

לכל טבלה בבסיס הנתונים, חייבים ליצור אובייקט בקוד שמייצג אותו – בדומה ל-NHibernate.

דבר נוסף, מיקרוסופט הצהירו שהם משחררים גרסה חדשה של Dot Net Core יחד עם EF Core בכל נובמבר. לכן, כבר יצאה גרסה 7, בנובמבר האחרון.  
בכל גרסה חדשה הם מביאים חידושים ושיפורים משמעותיים ולכן הם כיום אחת הספריות הטובות ביותר שקיימות בשוק כמעט בכל תרחיש אפשרי.

**[שקופית 6]**

אני אזכיר שנתוני כל טבלה מיוצגים ע"י אובייקט בקוד. כאשר, EF Core מנהל עבורנו מעקב אחרי הסטטוס של כל ישות אל מול בסיס הנתונים. כלומר, לכל אובייקט, יכול להיות סטטוס אחד מבין חמשה סטטוסים. אני אתייחס לכולם.  
הראשון, **Detached**, מציין שאין מעקב על האובייקט.

השני, **Unchanged**, מציין שהאובייקט קיים בבסיס הנתונים ולא בוצע עליו כל שינוי.  
השלישי, **Added**, מציין שהאובייקט לא קיים בבסיס הנתונים – ובפעולת שמירה תתבצע, למעשה, פעולת Insert.

הסטטוס הרביעי, **Modified**, מציין שהאובייקט קיים בבסיס הנתונים אבל הוא השתנה.

הסטטוס החמישי והאחרון, **Deleted**, מציין שהאובייקט קיים בבסיס הנתונים אבל הוא מסומן כ-"צריך להימחק" – ובפעולת שמירה תתבצע פעולת delete.  
הסטטוס הראשון, Detached, הוא היחיד שבו לא קיים מעקב על האובייקט.  
למעט פעולת שליפה – כלומר Read, כל הפעולות מול בסיס הנתונים יתבצעו בפועל ע"י קריאה לפונקציה בשם SaveChanges שקיימת בספרייה.

**[שקופית 7]**

מתי לא ניתן או לא נכון להשתמש ב-EF Core?  
השימוש ב-EF Core לא תמיד אפשרי. למשל, אם יש שימוש בטכנולוגיה שלא נתמכת - כמו פרויקט שכתוב ב-Dot Net Framework 4 הישן או בשביל לגשת לסוג בסיס נתונים שאינו נתמך – כמו Elastic Search.  
  
נקודה שכדאי לציין, למרות שהיא לא סיבה אמיתית מדוע לא להשתמש ב-EF Core. במקרים בהם מערכות צריכות לבצע מאות אלפי פעולות של הוספה או עדכון או מחיקה של רשומות בשאילתה אחת – מה שנקרא Bulk Operation. אז, יתכן ותהיה איטיות כאשר לא מנטרלים את המעקב אחרי הסטטוס של האובייקטים בשאילתה – כלומר אם יש Tracking. אפשר להגדיר שלא יהיה מעקב אחרי הסטטוס של האובייקטים בשאילתה שמריצים, זה מאוד פשוט – קוראים לפונקציה באותה שאילתה וזהו. אבל, למי שרוצה, יש nuget package שאפשר להתקין, שמבצע פעולות Bulk במהירות – כנראה ע"י נטרול המעקב אחרי האובייקטים.

**[שקופית 8]**

טוב, אחרי שדיברנו על השאלה מתי לא ניתן להשתמש ב-EF Core והבנו שזו הטכנולוגיה הכי טובה בעולם, אנחנו רוצים להתחיל לעבוד איתה. אז איך עושים זאת?  
מתקינים את ה-packages הנדרשים, יוצרים DbContext ומגדירים אותו – מה שמחליף את session ב-NHibernate – אנחנו נראה בקוד כמה שזה פשוט.

אז, לא משנה באיזו ספריה נבחר להשתמש, אנחנו חייבים ליצור אובייקט לכל טבלה.  
בנוסף, הספרייה חייבת להכיר את הטבלאות בבסיס הנתונים. ב-NHibernate עשינו זאת ע"י יצירת קבצי hbm. ב-EF Core אין דבר כזה.  
אז כיצד EF Core מכיר את בסיס הנתונים? – יש שלוש דרכים – לכל אחת מהן נראה שקופית בהמשך.

**[שקופית 9]**

**הדרך הראשונה**, אם לא עושים כלום, אז EF Core מתבססת על השמות. כלומר, שם הטבלה יקבע לפי שם ה-class של המודל, שדה Primary Key יקבע לפי שדה בשם Id ועמודות הטבלה וסוג הנתונים שלהם יקבעו אוטומטית לפי השדות של המודל.

ברור שאי אפשר באמת להסתפק בזה ולכן לא נבזבז זמן ונעבור לשקופית הבאה.

**[שקופית 10]**

**הדרך השנייה**, ע"י שימוש ב-Attributes על המודל והשדות שלו נוכל לציין במפורש את רוב ההגדרות הקיימות. למשל, לציין את שם הטבלה ב-database, להגדיר Primary Key, לציין את שמות העמודות בטבלה, להגדיר מפתח זר, אינדקסים, אורך מקסימאלי ועוד...

מצד אחד, אני מעדיף לא להתעכב על המצגת עוד הרבה זמן בגלל שאני רוצה להראות פרויקט רץ ועובד עם EF Core.  
אבל, מצד שני, ממש בא לי להתעכב עוד רגע על ה-Attribute-ים Timestamp ו-ConcurrencyCheck. שניהם משרתים לאותה מטרה – למנוע עדכון של אובייקט בבסיס הנתונים אם הערך שבעמודה המסומנות השתנה. בעזרתם אפשר למנוע עדכון של רשומה לא עדכנית – כלומר רשומה שהתעדכנה ע"י מישהו אחר לאחר שנשלפה.

**[שקופית 11]**

**הדרך השלישית** לגרום ל-EF Core להכיר את ה-database, שהיא גם הדרך המועדפת עלי, היא ע"י כתיבת קוד ייעודי. את הקוד אפשר לכתוב בשני מקומות לבחירתנו:  
או בתוך ה-DbContext או במחלקה נפרדת – אנחנו תכף נראה בפרויקט שזה לא מורכב בכלל.  
רק אציין, שהסיבה שבגללה אני מעדיף את הדרך הזאת היא שאפשר להגדיר בה כל הגדרה שנרצה לעומת הדרך השנייה שיש לה כמה מגבלות.

**[שקופית 12** – דיבור חופשי**]**