SpacePark

Reser runt i rymden, vi behöver stanna till och parkera skeppet.. Detta sker i en SpacePort.
SpacePort kontrolleras av SpacePark (lite som Göteborgs parkering).
Vi ska utveckla åt SpacePark.

Vi ska:

- Registrera parkeringar (Namn på resenär, skepp, vilken planet resenären kommer ifrån?)
- Öppna/stänga baserat på om det är fullt eller inte. Vi kan bara öppna åt skepp som det finns plats för
- Alla parkeringar ska registreras i en databas med Entity Framework Core (SQL?) Code first. Alla queries till databasen ska göras med Entity Frameworks fluent API.
- ENDAST kända Starwars karaktärer får parkera på denna spaceport.
  Dessa identifieras när de anländer till spaceporten, de ska betala innan de kan lämna parkeringen, och sist ska de kunna få ett ”kvitto”. (*TimeTable som räknar ut hur länge personen har stått där samt hur mycket som ska betalas?)*
- Vi ska hämta data med hjälp av REST API: https://swapi.dev/, Vi måste requesta API varje gång vi vill hämta data, dessa request ska göras med async.
- En rekommendation är att använda https://restsharp.dev/ detta NuGet paket. Läs på om denna via länken.
- Resenärerna kan endast använda STARSHIPS som de kan använda. Se swapapi.
- The travlers only use starships which have been part of a Starwars movie (see the endpoint /starships).
- They should be able to select their starship on arrival in the application.
- Samtliga requests ska vara asynkronoma

Som jag förstår det:

1. Menyval vid start av konsol: två alternativ, det ena är ”Register new parking” **(Se punkt 2-7)** och den andra är ”End current parking” (**se punkt 8)**.

2. Låt användaren knappa in ett namn

3. Anropa API med Async och kolla sedan om det matchar med någon ”people” i starwars API’t. (Detta kan göras exempelvis genom att endast anropa ”people/” så att vi får ut en count. Sedan loopar vi igenom för att se om namnet som användaren knappat in matchar med någon av de som finns i API’t.

4. Kolla så att SpacePark parkeringen inte är full, om den inte är full så gå till nästa steg

5. Om namnet matchar med det användaren knappat in, anropa då API för just den personens startships och ge hen ett menyval (som i Jakobs kurs) där den får välja vilken spaceship som ska parkeras. *Kanske ha en Console.Writeline(”Welcome Luke Skywalker, which spaceship are you docking?”)*

6. När Användaren valt ett skepp så måste vi sedan kolla så att det finns plats för skeppet, kanske kan hämta längden på skeppet i API’t? Då behöver vi isåfall sätta en gräns på hur långt ett skepp får vara. Om det finns plats så hoppa till nästa steg.

7. Om skeppet får plats samt att resenären är en Starwars karaktär (alltså om alla steg ovan gått igenom), så behöver vi registrera dessa i databasen.

8. Lägg in ett slutpris i databasen för den parkeringen.

TÄNKA PÅ:

Vad ska vi ha för tabeller i databasen? Vad ska de innehålla? Hur ska de vara kopplade till varandra?

Ska vi kunna hämta en ”saved” parkering?

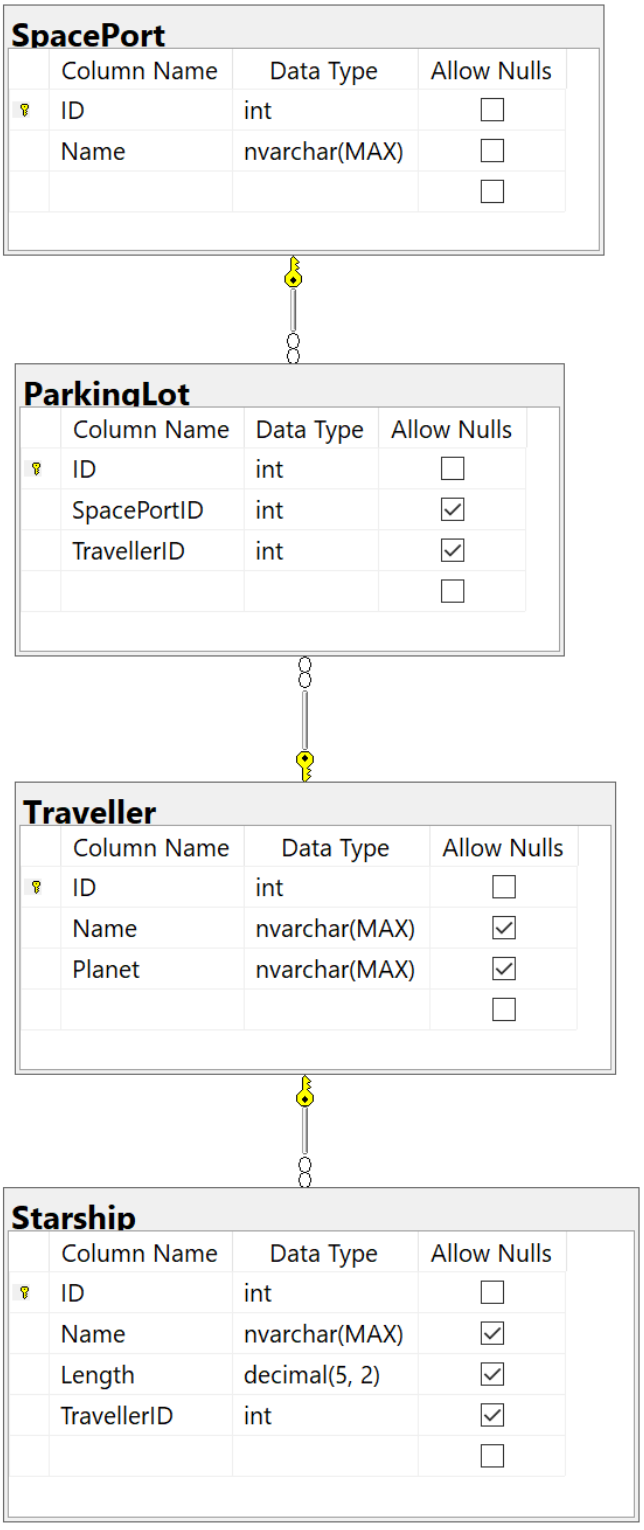
Ska vi beräkna priset på global datetime? Kanske 10kr/min bara för att demonstrera

Vi kommer behöva hämta: **people, planet, starship, station?** Som API.

Hur ska vi räkna på vad som får plats i stationen? Räknar vi på Length eller passenger? Eller hur gör vi här?

Vi behöver få hem dessa API: people: name, planets(optional?) | starships: name, length

Table demonstration EXAMPLE



Vi behöver även lägga till en Invoice eller Receipt tabell.