Tenta Avancerad Programmering

Daniel Lundholm – IT Högskolan 2020

# Refferenser

## Underlag

Underlag och kravdokument för projektet.

|  |  |
| --- | --- |
| Labb 1: Webb frontend och backend | <https://www.ithsdistans.se/pluginfile.php/33370/mod_resource/content/1/Labb%201.pdf> |
| Labb 2: Microservices | <https://www.ithsdistans.se/pluginfile.php/33603/mod_resource/content/1/Labb%202.pdf> |
| Labb 3: GDPR och cybersäkerhet | <https://www.ithsdistans.se/pluginfile.php/33985/mod_resource/content/1/Labb%203.pdf> |

## SCRUM

Trello board för projektets SCRUM planering: <https://trello.com/b/VuFUyJfh/tenta-avancerad-programmering>

## Git

Github remote repository för projektet: <https://github.com/Bragde/TentaAvanceradProgrammering>

# Projektbeskrivning

## Labb 1

Projektet ska vara en lite e-handel för (i ett första skede) dataspel. Det ska finnas en sida som listar alla spel. Sidor för registrering och inloggning. Möjlighet att lägga till spel i en varukorg och sedan bekräfta sitt köp.

**Teknisk beskrivning:** Användardata hanteras med Identity och Entity Framework. Varukorg och Ordrar lagras i minnet för tillfället.

## Labb 2

Lab 2 har samma funktionalitet som lab 1, men i lab 2 har jag brytit ut produktkatalogen, varukorgen och orderhanteringen från ”huvudprogrammet” och lagt dessa i varsitt eget projekt som en microservice. Varje microservice har en egen databas och ett tillhörande xUnit testprojekt. Alla databaser använder SQL Server och Entity Framework.

Det tidigare ”huvudprogrammet” återstår med funktionaliteten som ett användargränssnitt som använder sig av de olika microservicearna och syr ihop data från dessa där det behövs. Användarhanteringen (Identity) ligger i dagsläget fortfarande kvar i ”huvudprojektet” pga. tidsbrist, men tanken är att även detta ska lyftas ut till en microservice.  
Alla de tidigare funktionerna för in memory hantering av data för produkter, varukorg och ordrar har tagits bort ur ”huvudprogrammet” och hanteras av respektive microservice.

Testerna är skapade i xUnit. Jag har använt två olika metoder för testerna. I den ena metoden testar jag repository funktionerna direkt genom att använda Entity Framework in memory database och skapa en db context till en in memory test databas och verifiera data som sparas/ hämtas från databasen.  
I den andra metoden testar jag controllernas end-points genom att använda ASP.NET Core.TestHost och skapar en test klient för att göra http anrop till microservicarnas controllers, för att sedan verifiera responsens statuskoder från anropet.

## Labb 3

**Privacy policy:** När användaren loggar in på sajten första gången visas ett meddelande om att sajten använder cookies där användaren får möjlighet att acceptera detta, samt länk till mer information om vad detta innebär.

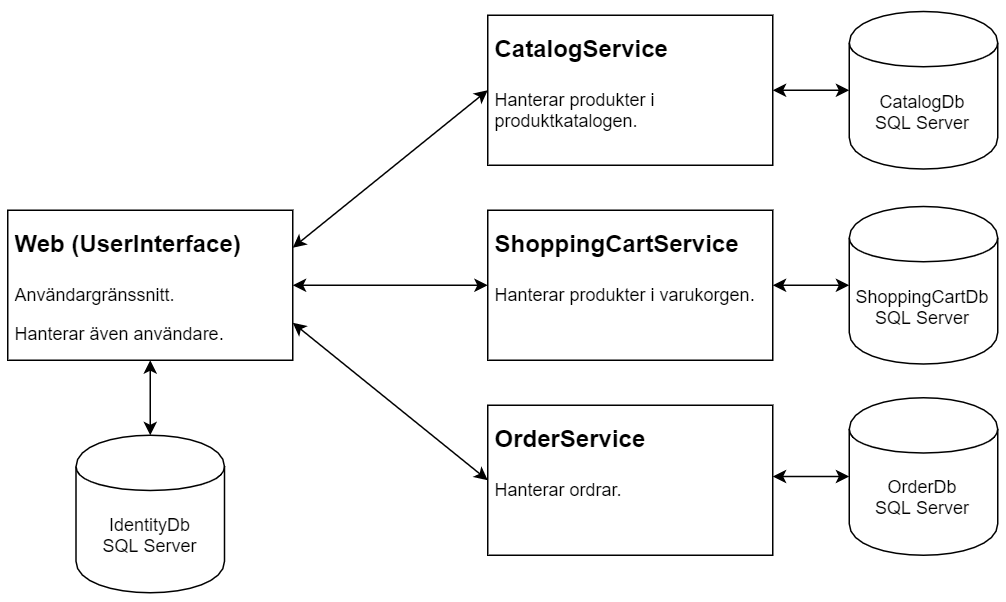
En sida har skapats med detaljerad information om sajtens användande av cookies och insamling av personlig information och data.

**Säkerhet:** Nedan listas vilka åtgärder som har tagits för att öka sajtens säkerhet.

* Alla microsevices kräver autentisering så att de endast kan användas av andra applikationer med korrekt nyckel.
* Entity Framework lagrar data i krypterat format (encrypted at rest) så att datan endast kan läsas av behöriga applikationer.
* Entity Framework använder Linq för att kommunicera med databasen vilket gör att en användare inte kan mata in egna SQL kommandon (SQL injection).

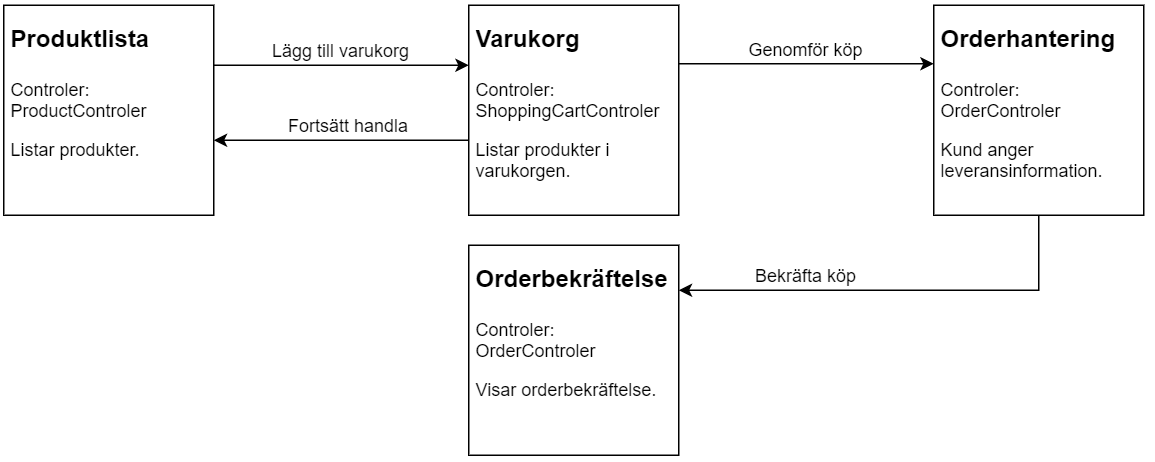
## Projekt struktur

Bilden visar hur projektets struktur är uppbyggd med användargränssnitt, microservices och databaser.



## Användar flöde

Visar hur saiten är uppbyggd och hur användarflödet är tänkt. Övriga funktioner än de som är visade finns och andra användarflöden är möjliga, men detta är det generella tänkta flödet.



## Funktionellt flöde

Nedan följer en beskrivning hur programmet arbetar med ovan beskrivna användarflöde som exempel.

* När programmets startas visas en sida som listar alla produkter. Om en användare är inloggad kan denna välja en produkt i listan att lägga till i varukorgen.
* Programmet lägger till produkten i varukorgen och visar varukorgen. Om produkten redan fanns i användarens varukorg plussas antalet på med 1, ny subtotal pris för produkten och totalt pris för varukorgen räknas ut.
* Användaren kan nu välja att fortsätta handla och kommer då tillbaka till produktlistan, eller att genomföra köp. Vid genomför köp hämtar programmet användarens information och visar en sida där användaren kan kontrollera denna information och sedan välja att bekräfta köpet.
* Vid bekräftat köp skapar programmet en order för användaren med produkterna i varukorgen och en ny bekräftelse sida visas med information om ordern. Programmet tar också bort användaren varukorg.