Frontend 2 – Logg

12/01-2026

Importerar koden från javascript, html o css till ett react projekt. Väljer react då jag har mest erfarenhet av den.

13/01-2026

Skapar att man kan toggla mellan ledig och vald sittplats och även skapar object med arrayer som reprenterar sitt platserna och renderas till UI med hjälp av map metod som retunerar sittplats-komponent. Ska använda mig av useReducer så mycket som möjligt då jag vill lära mig Redux längre fram och då det är lämpligt att använda sig av när man har många states.

14/01-2026

Hämtar filmer från API (C#). Rendera datan från API och mappar det till en klass innan det renderas i en dropdown meny. (Vid det här tidspunkten så visste jag inte att man skulle göra en json-server 28.01.2026).

15/01-2026

Gjort om/utökad useReducer. Gjort lite API och lista av sittplatser i C# (Test bara). La till URL för fetch. Skapade komponent för dropdown-menyn.

16/01-2026

Utökat min useReducer med nya keys.

17/01-2026

Implementerat så att det automatiskt väljer en film åt en så fort man går in på sidan om det skulle vara så att man vill direkt klicka på sittplatserna om det råkar vara den filmen man vill se. Stötte på problemet att den inte väljer den första filmen då jag la till att på onChange på den selecten. Så bara när man man ändrar select är då man kan välja. Så jag löste det genom att när man väl fetchar datan och lägger in det i en lista, redan där ska man lägga till att man har "valt" den första filmen från "databasen" och pengarna korrigeras efter filmen * hur många sittplatser man har valt.

20/01-2026

Här ville jag nästan implementera att sittplatserna skulle också från api:et för att försöka koppla sittplatserna med besökarna eller filmerna och spara de i localstorage eftersom jag tänkte då att det inte fins en databas men så småningom skrotade jag den idén, eller pausade den (28.01.2026)

22/01-2026

Här implementerade jag min slutgiltiga klass implementering då på råd av chatGPT skulle jag göra innan map.metoden. Och började med min AdminPanel, körde React Router, la mest tid på att reda ut hur man gjorde NavLink osv. Tänkte att det var lämpligast att man ska göra det på en hel annan sida än där besökarna bokar sina platser.

23/01-2026

Började med att göra individuella CRUD komponenterna för AdminPanel. Implementerade fetch logik i AdminPanel för att sedan för detta vidare till CRUD-barnen och satte igång med CSS på navbaren där man ska välja metoden man vill köra.

24/01-2026

Gjorde snygg button som passar med hemsidans tema. Implementerade automatisk val (den första på listan som fetchas in) på put samt logic för uppdatera i put komponenten och radera för delete komponenten.

25/01-2026

Jag gjorde så att när man kommer in i och trycker på GET så ska det dyka upp 2 buttons till och beroendepå vilket man väljer så ska man antingen få en lista på alla filmer eller lägga in namnet på en film och sedan visar en film så att man gör en GET request på både hela listan och om man vill ha en film. Implementerade logik för Post.jsx.

26/01-2026

Här började jag implementera json-server, installderade json-server på min package.json och bytte api url från C# till json-serverns url på alla kompoenenter i min AdminPanel.

27/01-2026

Arbetade med refaktorering och förbättring av bokningsflödet i applikationen. Uppdaterade datatyper (bl.a. pris från sträng till nummer), anpassade ID-hantering för json-server och rensade bort redundant kod. Förbättrade state-logik i `useReducer`, lade till validering för formulär och sätesbokning samt färdigställde bokningslogiken. Uppdaterade API-anrop, förbättrade rendering (keys i map, villkorlig rendering) och justerade styling med ny CSS för knappar och inputs.

28/01-2026

Deplayot mitt repo till Github Pages. Fick igen problemet att sidan inte syns och Ctrl + F5 levererar goda resultat återigen!

Användning av json-server

I uppgiften var json-server ett krav, och därför användes det som backend-lösning. Kravet innebar att applikationen skulle kommunicera med ett API och kunna utföra CRUD-operationer, utan att behöva implementera en fullständig backend eller databas.

Json-server uppfyllde dessa krav genom att tillhandahålla ett enkelt REST-API där filmer och bokningar kunde lagras och hanteras via HTTP-anrop. Det gjorde det möjligt att fokusera på frontend-logik, formulärhantering och state-hantering i applikationen, istället för att lägga tid på backend-setup som inte efterfrågades i uppgiften.

En riktig backend hade tillfört mer setup, mer kod och mer komplexitet som inte efterfrågades i kraven, utan hade istället tagit fokus från det som uppgiften faktiskt handlade om.

Val av React istället för vanilla JavaScript

Jag valde React eftersom jag har erfarenhet av ramverket och eftersom det gör det enklare att hantera ett dynamiskt användargränssnitt. Applikationen innehåller flera delar som förändras över tid, till exempel vald film, valda sittplatser och om bokningsformuläret ska visas eller inte.

I vanilla JavaScript hade detta krävt betydligt mer kod där varje förändring i state måste hanteras manuellt steg för steg. Med React kan jag istället arbeta mer deklarativt, där UI:t automatiskt uppdateras när state ändras. Det innebär att jag i större utsträckning beskriver när något ska visas, istället för exakt hur varje uppdatering ska göras.

Detta gjorde koden mer överskådlig och lättare att underhålla, särskilt när flera komponenter påverkas av samma state.

Struktur och mappindelning

Projektet är uppdelat i komponenter för att hålla koden organiserad och lätt att förstå. Admin-funktionaliteten ligger i en egen mapp för att tydligt separera användargränssnittet för besökare från administrationsfunktioner som CRUD-operationer på filmer.

Denna struktur gör projektet mer lättläst och skalbart. Om projektet skulle bli större i framtiden finns redan en tydlig uppdelning som gör det enklare att lägga till fler funktioner, komponenter eller vyer utan att koden blir rörig.