Enor Zilkiqi & Nasser Al-Obaedi
2ME303 - Verksamhetsförlagd praktik i medieteknik
Linnéuniversitetet - 2025-01-13

Linnéuniversitetet

Troja kund & mailsystem

Författare & Utvecklare: Enor Zilkiqi (GitHub) & Nasser Al-Obaedi (GitHub)

Inledning

Syftet med denna rapport är att ge en övergripande blick över hur Trojas specialanpassade kund & mailsystem är uppbyggt samt hur alla separata delar och funktioner samverkar med varandra. Detta görs för att underlätta vidareutveckling för andra utvecklare som eventuellt tar över och vidareutvecklar denna tjänst.

Rapporten delas in i två delar, frontend och backend, för att beskriva dess struktur.

Översikt

Projektets ändamål var att effektivisera arbetsflödet för administrationen på hockeyklubben Troja Ljungby vad gäller kundhantering och mailutskick samt att implementera ett kundlojalitetsprogram, något som de tjänster Troja använder i dagsläget inte erbjuder.

Med denna specialanpassade webblösning som beskrivs i denna rapport ges nu möjlighet för Troja att ha kunder och mailutskick samlade i ett och samma system. Systemet är en basprodukt (första version) på ett system som kan vidareutvecklas enligt Trojas vision och behov.

Systemet bygger kring ett antal API:er, främst Tickster för all kund- och försäljningsdata men även Resend (e-posttjänst) och Beefree.SDK (e-postbyggaren).

De teknologier som används vid utvecklandet av systemet är:
- Express.js (backend)
- MySQL (backend/databas)
- React (frontend)

Frontend

Sidor

Systemet består sammanlagt av 5 huvudsidor som tillsammans utgör hela tjänsten. Sidorna i sig är uppbyggda av olika komponenter för rendering av data och funktionalitet. Nedan beskrivs sidorna och dess syfte:

- **AdminLogin**
    - Ansvarar endast för inloggning till tjänsten och förhindrar obehöriga från åtkomst.
        - Innehållande komponenter: LoginContainer, LoadingCircle.
        - Innehållande services: loginAdmin.
- **Home**
    - Landningssidan man kommer till efter att man loggat in. Denna sida ger användaren information om bland annat nästkommande matcher, kund- och försäljningsstatistik.
        - Innehållande komponenter: StackGrid, InfoContainer.
- **Customers**
    - Sidan som huserar alla kunder i en tabell. Tabellen går att filtrera enligt olika kriterier för att leta fram specifika kunder, samt sortera kunderna i stigande eller fallande ordning. Även specifik kundinformation är tillgänglig här.
        - Innehållande komponenter: CustomerTable, Toolbar, CustomerFilter, Button, LoadingCircle.
        - Innehållande services: triggerFetchData.
- **Mailing**
    - Denna sida ansvarar för mailutskick till kunderna. Här finns e-postbyggarkomponenten där administrationen har möjlighet att bygga sina egna e-postmallar och skicka dessa till respektive kundgrupper. Mallarna går

att sparas lokalt och även laddas in i byggaren från klienten för återanvändning.
    - Innehållande komponenter: Toolbar, EmailBuilder, Button, SelectCustomerGroupModal, LoadingCircle, SubjectModal.
    - Innehållande services: fetchCustomersGroupedByGoods
- **Settings**
    - En sida för inställningar. Här kan man ändra användarnamn och lösenord till tjänsten.
        - Innehållande komponenter: Toolbar, SettingsContainer, AvatarIcon.

Komponenter

Beskrivning

Nedan kommer en beskrivning av komponenternas syfte och funktionalitet.

- **AvatarIcon**
    - Enkel komponent som visar Trojas logga som ett användarkonto på inställningssidan.
- **Button**
    - Generell knapp-komponent som återanvänds på flera ställen i källkoden.
- **CustomerFilter**
    - Komponenten som har hand om sök filtreringen på kundtabellen. Sätter sökkriterium och värde. Customers sidan hanterar staten där CustomerFilter sköter input och värdet skickas till CustomerTable via Customers för rendering.
- **CustomerModal**
    - Komponenten som visar specifik kundinformation när man klickar på en kund i tabellen. Modalen gör ett anrop till servern om specifik kundinformation med variabeln customer.userRefNo.
- **CustomerTable**
    - Kundtabellen som innehåller alla kunder. Denna komponent importerar direkt alla kunder som finns på den egna databasen och renderar dem för användaren. Tabellen tar emot parametrarna searchQuery och searchCriteria för att filtrera resultaten enligt användarens önskemål.
- **DropdownSelect**
    - Generell komponent för en dropdown komponent

- **Emailbuilder**
    - E-postbyggarkomponenten som instansierar Beefree.SDK-paketet med hjälp av miljövariabler. E-postbyggaren konfigureras i denna komponent och ytterligare funktionalitet läggs till för att möjliggöra sparandet av e-postmallar lokalt på hårddisken i JSON-format. Komponenten hanterar även att skicka e-postmallar i HTML-format.
- **InfoContainer**
    - En behållarkomponent för diverse information på "Hem"-sidan. Hämtar nästkommande matcher genom Tickster-api:et samt visar biljettstatistik (hämtas från lokala databasen) och senaste e-post rubriker.
- **InputField**
    - Generell komponent för inmatningsfält som återanvänds på olika ställen i programmet.
- **LoadingCircle**
    - Generell komponent för laddningsanimation. Används när asynkrona funktioner är aktiva för att ge användarfeedback.
- **LoginContainer**
    - Komponenten som hanterar användarinloggning. Skickar inmatning av användarnamn och lösenord till den lokala databasen för verifiering.
- **MaintenanceClock**
    - Komponent som inte används men finns kvar för eventuell användning. Var tänkt att användas som en timer för när den automatiska synkroniseringen (Cron arbetet) av den lokala databasen mot Tickster skulle ske.
- **NavBar**
    - Komponent som sköter navigeringen mellan de olika sidorna i systemet.
- **SelectCustomerGroupModal**
    - En komponent som låter användaren välja kundgrupper för e-postutskick. Den visar en paginerad tabell med kundgrupper och hanterar sökning och val av grupper. Valda grupper sparas i localStorage och återställs vid laddning. Grupperna som väljs regelbundet visas först i tabellen för enklare åtkomst.
- **SettingsContainer**
    - En komponent som hanterar användarens kontoinställningar. Användaren kan ändra sitt användarnamn och lösenord.
- **StackGrid**
    - En komponent som hämtar och visar försäljningsdata och kundstatistik. Den filtrerar bort köp med nollpris och tar bort dubbletter baserat på crmId. Komponenten beräknar total försäljning och identifierar den kund som spenderade mest under föregående månad.
- **SubjectModal**

    - En modal som hanterar ämnesval för e-post. Användaren kan skriva in ett nytt ämne eller välja ett tidigare sparat ämne från en lista. Sparade ämnen lagras i localStorage och återställs vid laddning. Komponenten tillåter även att rensa alla sparade ämnen.
- **Toolbar**
    - En komponent som fungerar som en behållare för verktygsfältet. Den omsluter sina barnkomponenter (children) i en Material-UI Paper-komponent med en Box-layout. Komponenten används för att strukturera och visa verktyg och kontroller på olika sidor i applikationen.
- **UpcomingGame**
    - En komponent som visar information om en kommande match. Den tar emot matchdata som props och formaterar datum och tid för visning. Komponenten delar upp matchnamnet för att få hem- och bortalag, och hämtar respektive laglogotyper från TeamLogos (en utility fil). Den renderar matchens namn, datum, tid och lag logotyper i en strukturerad layout.

Utils

En mapp som innehåller hjälpfunktioner och konfigurationsfiler som används över hela applikationen.

- **MaterialUI**
    - Importerar Material-UI-komponenter och konfigurerar temat på dessa komponenter som används i applikationen.
- **TeamLogos**
    - Ett objekt som mappar lagnamn till deras respektive logotyper. Används för att dynamiskt hämta och visa lag logotyper i komponenter som UpcomingGame.
- **ProtectedRoute**
    - En komponent som skyddar vissa rutter i applikationen. Den kontrollerar om en giltig token finns i localStorage. Om token finns, renderas den angivna komponenten (Component). Om ingen token finns, omdirigeras användaren till inloggningssidan (/login).

Services

En mapp som innehåller tjänstefiler som hanterar API-anrop för applikationen. Dessa tjänster används för att kommunicera med backend och hämta eller skicka data.

- **adminService**
    - En tjänst som hanterar API-anrop relaterade till administratörsinloggning.
- **customerService**
    - En tjänst som hanterar API-anrop relaterade till kunder. Innehåller funktioner för att:
    - Hämta alla kunder (fetchAllCustomers)
    - Hämta kunder grupperade efter köpta varor (fetchCustomersGroupedByGoods)
    - Hämta köp för en specifik kund (fetchCustomerPurchases)
- FetchUpcomingEvents
    - En tjänst som hanterar API-anrop relaterade till nästkommande matcher och hämtar dessa från databasen.

Backend

App.js

Huvudinstegspunkt för backend-applikationen byggd med Node.js och Express.

Använder de aktuella modellerna.

- **Express**: För att skapa webbservern och hantera HTTP-förfrågningar.
- **CORS**: För att möjliggöra Cross-Origin Resource Sharing, vilket tillåter servern att acceptera förfrågningar från olika ursprung.
- **Logger**: För att logga meddelanden och fel till konsolen eller en fil.
- **Admin Routes**: För att hantera routes relaterade till administrativa funktioner.
- **Node-Cron**: För att schemalägga uppgifter att köras med specifika intervall.

Logger.js

Skapar en fil med hjälp av Winston för att logga eventuell information som loggas av servern. Den fanns, men användes knappt i projektet. På grund av tidsbegränsningar var det tänkt att implementeras senare.

Routes

Admin Routes

Beskrivning: routes för administratörsbehörigheter och datahantering.

POST /login
Funktionalitet: Denna route används för admininloggning. Den tar emot användarnamn och lösenord och autentiserar adminanvändaren.
Handler: adminLogin

POST /register
Funktionalitet: Denna route tillåter registrering av en ny adminanvändare. Den kräver autentisering (via authenticateToken) för att säkerställa att endast auktoriserade användare kan registrera nya admins.
Handler: registerAdmin

PUT /change-details
Funktionalitet: Denna route används för att uppdatera detaljerna för en befintlig adminanvändare (t.ex. användarnamn, lösenord). Den kräver autentisering för att säkerställa att endast den inloggade adminen kan ändra sina uppgifter.
Handler: changeAdminDetails

POST /trigger-fetch
Funktionalitet: Denna route triggar en datahämtning från Ticksters API och loggar det. Den kräver autentisering för att säkerställa att endast auktoriserade användare kan initiera denna operation.
Handler: triggerFetchData

POST /send-email
Funktionalitet: Denna route används för att skicka e-post för nyhetsbrev. Den kräver autentisering för att säkerställa att endast auktoriserade användare kan skicka e-post.
Handler: sendMail

Customer Routes

GET /customers
Funktionalitet: Denna route hämtar en lista över alla kunder från databasen. Den kräver

autentisering för att säkerställa att endast auktoriserade användare kan få tillgång till kunddata.
Handler: fetchAllCustomers

GET /customers/grouped-by-goods
Funktionalitet: Denna route hämtar kunder grupperade efter de varor de har köpt. Den kräver autentisering för att säkerställa att endast auktoriserade användare kan få tillgång till denna data.
Handler: fetchCustomersGroupedByGoods

GET /customer/:userRefNo/recent-purchases
Funktionalitet: Denna route hämtar de senaste köpen som en specifik kund, identifierad med userRefNo, har gjort. Den kräver autentisering för att säkerställa att endast auktoriserade användare kan få tillgång till denna data.
Handler: fetchRecentPurchases

GET /customers/last-month
Funktionalitet: Denna route hämtar antalet kunder som registrerade sig under den senaste månaden. Den kräver autentisering för att säkerställa att endast auktoriserade användare kan initiera denna operation.
Handler: fetchCustomersLastMonth

GET /customers/last-year
Funktionalitet: Denna route hämtar antalet kunder som registrerade sig under det senaste året. Den kräver autentisering för att säkerställa att endast auktoriserade användare kan initiera denna operation.
Handler: fetchCustomersLastYear

Event Routes

GET /events
Funktionalitet: Denna route hämtar en lista över alla evenemang från databasen. Den kräver autentisering för att säkerställa att endast auktoriserade användare kan få tillgång till evenemangsdata.
Handler: fetchAllEvents

GET /upcoming-events
Funktionalitet: Denna route hämtar en lista över kommande evenemang. Den kräver autentisering för att säkerställa att endast auktoriserade användare kan få tillgång till denna information.
Handler: fetchUpcomingEvents

Goods Routes

GET /goods
Funktionalitet: Denna route hämtar en lista över alla varor som finns tillgängliga i databasen. Den kräver autentisering för att säkerställa att endast auktoriserade användare kan få tillgång till varudata.
Handler: fetchAllGoods

Purchase Routes

GET /purchases
Funktionalitet: Denna route hämtar en lista över alla köp som gjorts av kunder. Den kräver autentisering för att säkerställa att endast auktoriserade användare kan få tillgång till köpdata.
Handler: fetchAllPurchases

GET /purchases/recent-with-details
Funktionalitet: Denna route hämtar de senaste köpen tillsammans med deras detaljer. Den kräver autentisering för att säkerställa att endast auktoriserade användare kan få tillgång till denna data.
Handler: fetchRecentPurchasesWithDetails

GET /customer-purchase-counts
Funktionalitet: Denna route hämtar antalet köp som varje kund har gjort. Den kräver autentisering för att säkerställa att endast auktoriserade användare kan få tillgång till denna data.
Handler: fetchCustomerPurchaseCounts

GET /purchases/total-revenue-by-recent-events
Funktionalitet: Denna route hämtar den totala intäkten som genererats av de senaste evenemangen. Den kräver autentisering för att säkerställa att endast auktoriserade användare kan få tillgång till denna data.
Handler: fetchTotalRevenueByRecentEvents