Software Design Document

Festival travel system (Web development 2024-2025)

Auteur: Jesper Meuzelaar

Reviewer: Tim van Dijk

Table of Contents

1.0 Algemeen Overzicht

1.1 Doel

In dit Software-design-document zal ik beschrijven hoe mijn, Travel system Applicatie in elkaar zit. Zodat de client het ontwerp begrijpt en er ook meer uitleg kan worden gegeven aan bepaalde keuzes. Ook Zal ik in dit SDD-inzicht geven in het technische ontwerp van de Applicatie.

1.2 Scope

De Travel System-applicatie biedt een breed scala aan functies om zowel klanten als beheerders optimaal te ondersteunen. Klanten kunnen zich registreren, hun gegevens beheren en busreizen naar festivals boeken via een overzichtelijk platform dat ook inzicht geeft in beschikbare reis- en festivaldata.

Het systeem bevat ook een loyaliteitspuntenprogramma waarmee klanten bij elke reservering punten kunnen verzamelen. Deze punten kunnen worden ingewisseld voor voordelen zoals kortingen en VIP-toegang, waardoor de klantloyaliteit wordt bevorderd.

Beheerders hebben uitgebreide mogelijkheden voor het beheren van festivalgegevens, waaronder het toevoegen, bewerken, bekijken en verwijderen van festivals. Klantbeheer geeft inzicht in klantgegevens en reisgeschiedenis.

Planners kunnen ook profiteren van automatische triggers die inschrijvingen analyseren en busreizen plannen als er meer dan 35 inschrijvingen zijn. Loyaliteitspuntenprogramma's zijn ook geïntegreerd in het besluitvormingsproces, en beschikbare punten kunnen de bus planning en het gebruik beïnvloeden.

De applicatie zal worden ontwikkeld met een modulaire aanpak die flexibiliteit en uitbreidbaarheid garandeert. Door de functionaliteit op te delen in afzonderlijke modules kan elk onderdeel afzonderlijk worden verbeterd en uitgebreid. Het databaseontwerp ondersteunt deze modules met een schema dat zowel klantgerichte als administratieve taken stroomlijnt zonder de prestaties of gegevensintegriteit in gevaar te brengen.

1.3 Overzicht

Dit Software Design Document (SDD) biedt een uitgebreide beschrijving van het ontwerp van de applicatie, zowel op technisch als visueel vlak. Het document is bedoeld als leidraad voor zowel ontwikkelaars als stakeholders om inzicht te krijgen in de structuur, werking en ontwikkeling van het systeem.

Het SDD is opgebouwd uit de volgende onderdelen:

- **Hoofdstuk 2 – Functionele Specificaties**
  Beschrijft de functionele en niet-functionele eisen waaraan het systeem moet voldoen. Deze specificaties vormen de basis voor het ontwerp en de implementatie.

- **Hoofdstuk 3 – Systeemarchitectuur en Technisch Ontwerp**
  Gaat in op de gekozen ontwikkelomgeving, de technische structuur van de applicatie, het datamodel en de gebruikte frameworks. Dit hoofdstuk legt de brug tussen de functionele eisen en de daadwerkelijke implementatie.

- **Hoofdstuk 4 – Interface Ontwerp**
  Toont de wireframes en licht de ontwerpkeuzes toe. Hierbij wordt ook de navigatie en gebruikersinteractie besproken.

- **Hoofdstuk 5 – Testplan**
  Beschrijft de teststrategie, gebruikte tools en criteria voor succesvolle oplevering van de applicatie.

- **Hoofdstuk 6 – Implementatie en Deployment**
  Geeft inzicht in de ontwikkelingsfasen, de gebruikte tools, en hoe de applicatie wordt uitgerold naar de productieomgeving.

2.0 Functionele / Niet functionele requirements

2.1 Functionele Requirements

Nr.	Functionele Requirement
FR-1	Gebruikers kunnen zich registreren met hun volledige naam, e-mailadres, sterk wachtwoord (minimaal 8 tekens) en geboortedatum.
FR-2	Ingelogde gebruikers hebben toegang tot een persoonlijk dashboard waar ze hun profielgegevens kunnen beheren en hun boekingsgeschiedenis kunnen bekijken.
FR-3	Gebruikers kunnen beschikbare festivals en bijbehorende busreizen bekijken en filteren op datum, locatie en genre.
FR-4	Gebruikers kunnen busreizen boeken voor een gekozen festival inclusief selectie van datum en opstapplaats.
FR-5	Na een succesvolle boeking ontvangt de gebruiker binnen 60 seconden een bevestiging via e-mail en worden er 10 loyaliteitspunten toegevoegd aan het account.
FR-6	Loyaliteitspunten kunnen bij een volgende boeking worden ingewisseld voor korting, via een optie in het boekingsproces.
FR-7	Beheerders kunnen via een beveiligd portaal festivals en klantaccounts beheren, en bussen handmatig toevoegen of aanpassen.

2.2 Niet functionele requirements

Nr.	Niet-Functionele Requirement
NFR-1	De database moet relationeel zijn en genormaliseerd volgens ten minste 3NF om redundantie te beperken en consistentie te waarborgen.
NFR-2	De database moet uitbreidbaar zijn, zodat nieuwe tabellen en relaties kunnen worden toegevoegd zonder bestaande functionaliteit aan te passen.
NFR-3	Boekingen moeten binnen 2 seconden verwerkt worden om dubbele reserveringen te voorkomen en directe feedback te geven.
NFR-4	Loyaliteitspunten moeten binnen 1 seconde na het voltooien van een boeking worden bijgewerkt en zichtbaar zijn op het dashboard.
NFR-5	De webinterface moet volledig responsive zijn en correct werken op schermformaten van 320px tot 1920px breed.
NFR-6	De applicatie moet een gebruiksvriendelijke UX bieden, met intuïtieve navigatie en directe visuele feedback bij gebruikersacties.
NFR-7	De backend moet onder een gemiddelde responstijd van 500ms blijven bij een belasting van 50 gelijktijdige gebruikers.
NFR-8	De applicatie moet bij Lighthouse-audits minimaal een score van 90 behalen op Performance, Best Practices en SEO.

3.0 Systeemarchitectuur en Technisch Ontwerp

3.1 Ontwikkelomgeving

Voor de ontwikkeling van deze webapplicatie is gekozen voor een moderne full-stack opzet, met een focus op schaalbaarheid, veiligheid en ontwikkelsnelheid. De volgende tools en technologieën zijn ingezet:

- **Frontend**: HTML, Tailwind CSS, JavaScript
- **Backend**: PHP met Laravel Framework
- **Database**: MySQL
- **Tools**: Visual Studio Code, Git & GitHub, Postman voor API-testen
- **Overige**: Composer (dependency management), Laravel Artisan CLI

De gekozen stack sluit aan bij de functionele eisen zoals authenticatie, databasekoppeling en routing, en maakt het mogelijk om zowel snel te prototypen als robuuste backendlogica te implementeren.

3.2 Ontwikkelproces

Tijdens de ontwikkeling is gebruik gemaakt van een versiebeheersysteem (Git), waarbij meerdere branches zijn ingezet om de stabiliteit van de hoofdtak te waarborgen. Belangrijke principes die hierbij gevolgd zijn:

- **Branching strategie**: feature branches per onderdeel (bijv. login, boeking, dashboard)
- **Pull requests & code reviews**: lokaal of via GitHub verwerkt

Daarnaast is er gewerkt met een iteratieve aanpak, waarbij per fase getest en aangepast werd. Hierdoor kon de feedback van gebruikers en docenten snel verwerkt worden in de ontwikkeling.

3.3 Productieomgeving

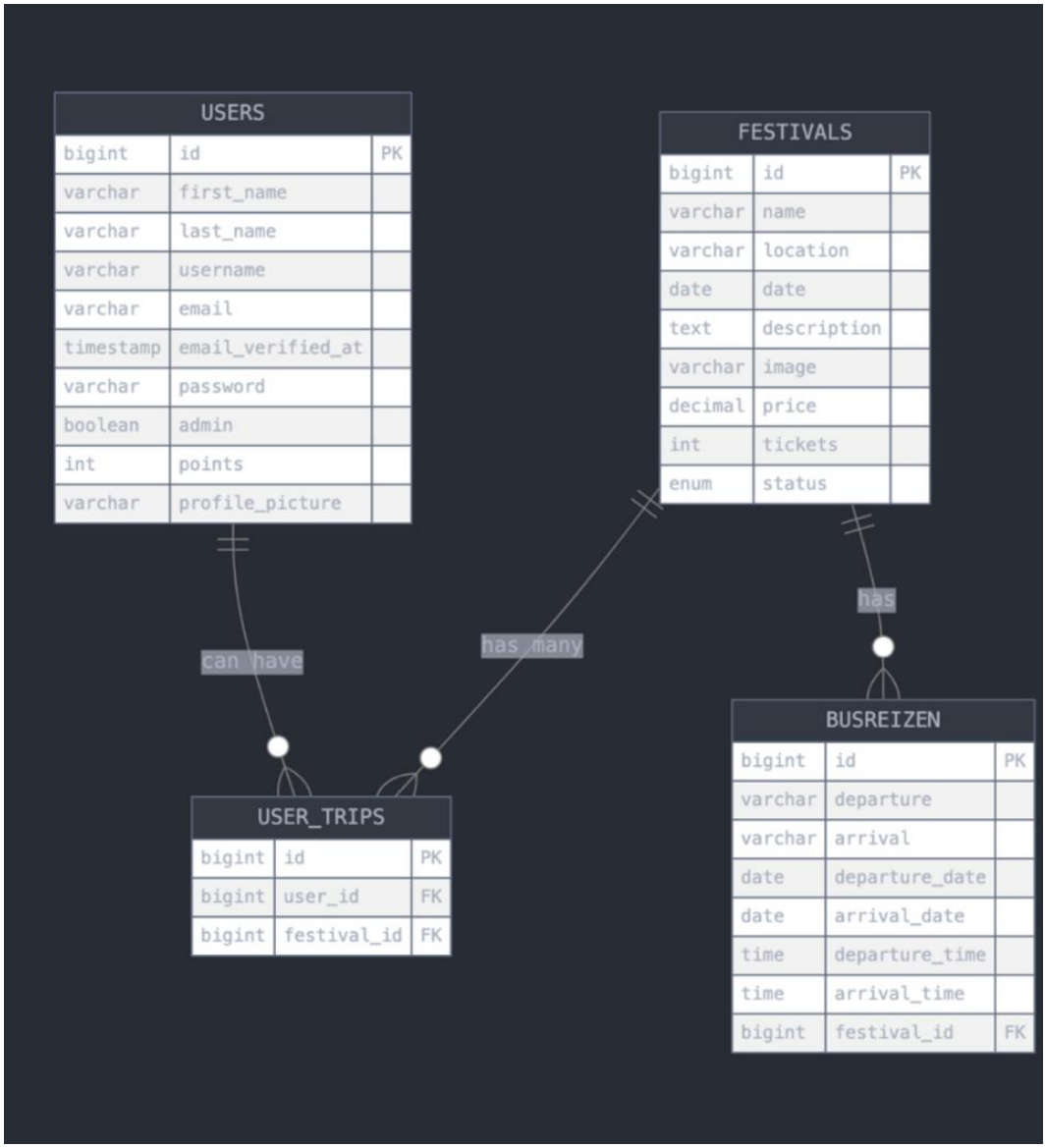
Tijdens dit project is de applicatie uitsluitend lokaal ontwikkeld en getest. Er is **geen live productieomgeving** ingericht, aangezien het project is uitgevoerd als oplevering binnen een onderwijscontext.

De lokale ontwikkelomgeving bestond uit:

- **Webserver**: Laravel's ingebouwde development server (php artisan server)
- **PHP-versie**: 8.x, compatibel met Laravel
- **Database**: MySQL, lokaal beheerd met mySQL workbench
- **Bestandsbeheer**: Codeversies zijn beheerd via Git op een lokale repository
- **Testing**: Zowel frontend- als backendfunctionaliteiten zijn lokaal getest via browser
- Er is ruimte om de applicatie in de toekomst uit te breiden naar een publieke omgeving, waarbij hosting op platforms zoals Laravel Forge, Vercel of een VPS mogelijk is.

3.4 Datamodel (ERD)

Hieronder is het **Entity-Relationship Diagram (ERD)** te zien van de applicatie. Dit model toont de belangrijkste entiteiten (gebruikers, boekingen, festivals, bussen, loyaliteitspunten, etc.) en de onderlinge relaties.

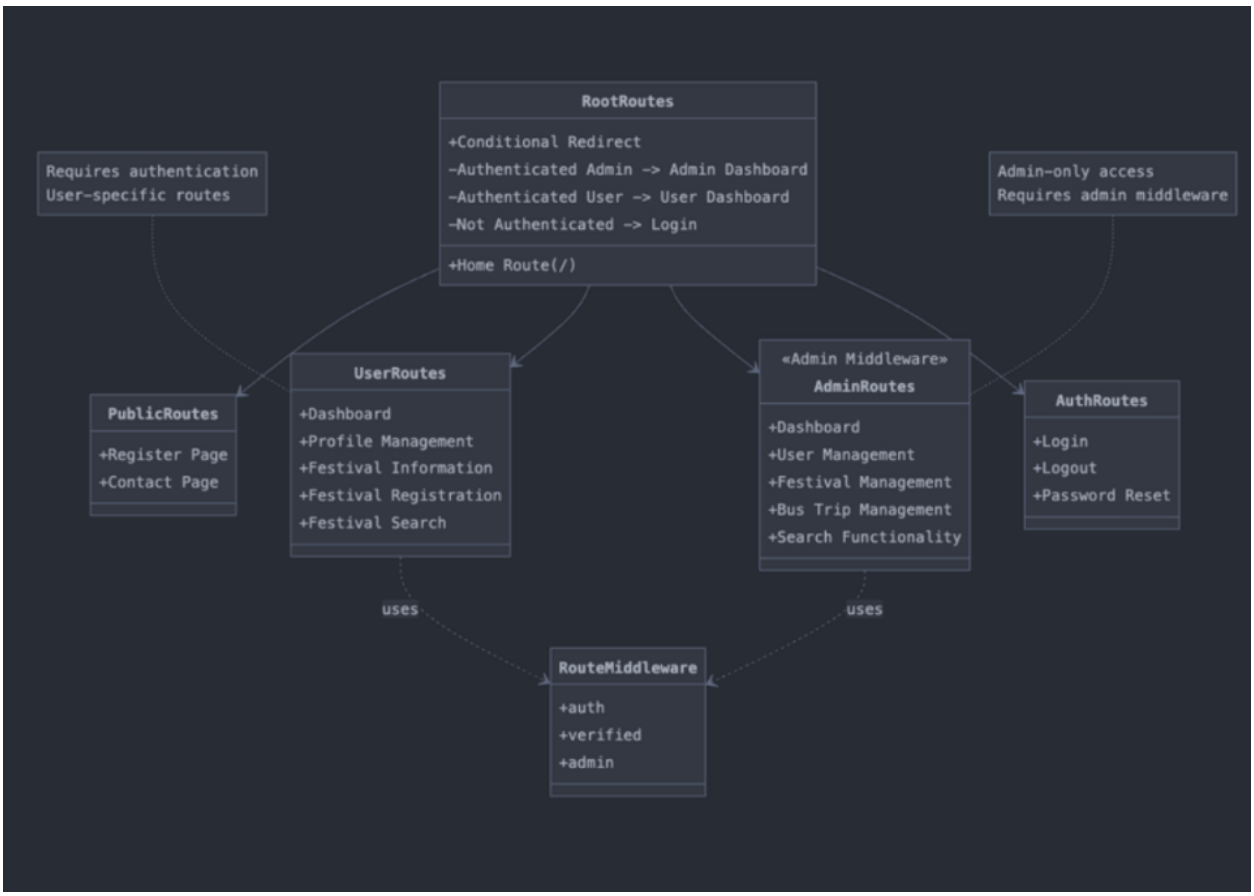


Toelichting:
Dit ERD-model beschrijft een systeem waarin gebruikers zich kunnen registreren voor festivals en daarbij optioneel busreizen kunnen boeken. Het model koppelt gebruikers en festivals via een tussenliggende USER_TRIPS tabel en verbindt busreizen direct aan festivals.

3.5 Architectuur en Logica

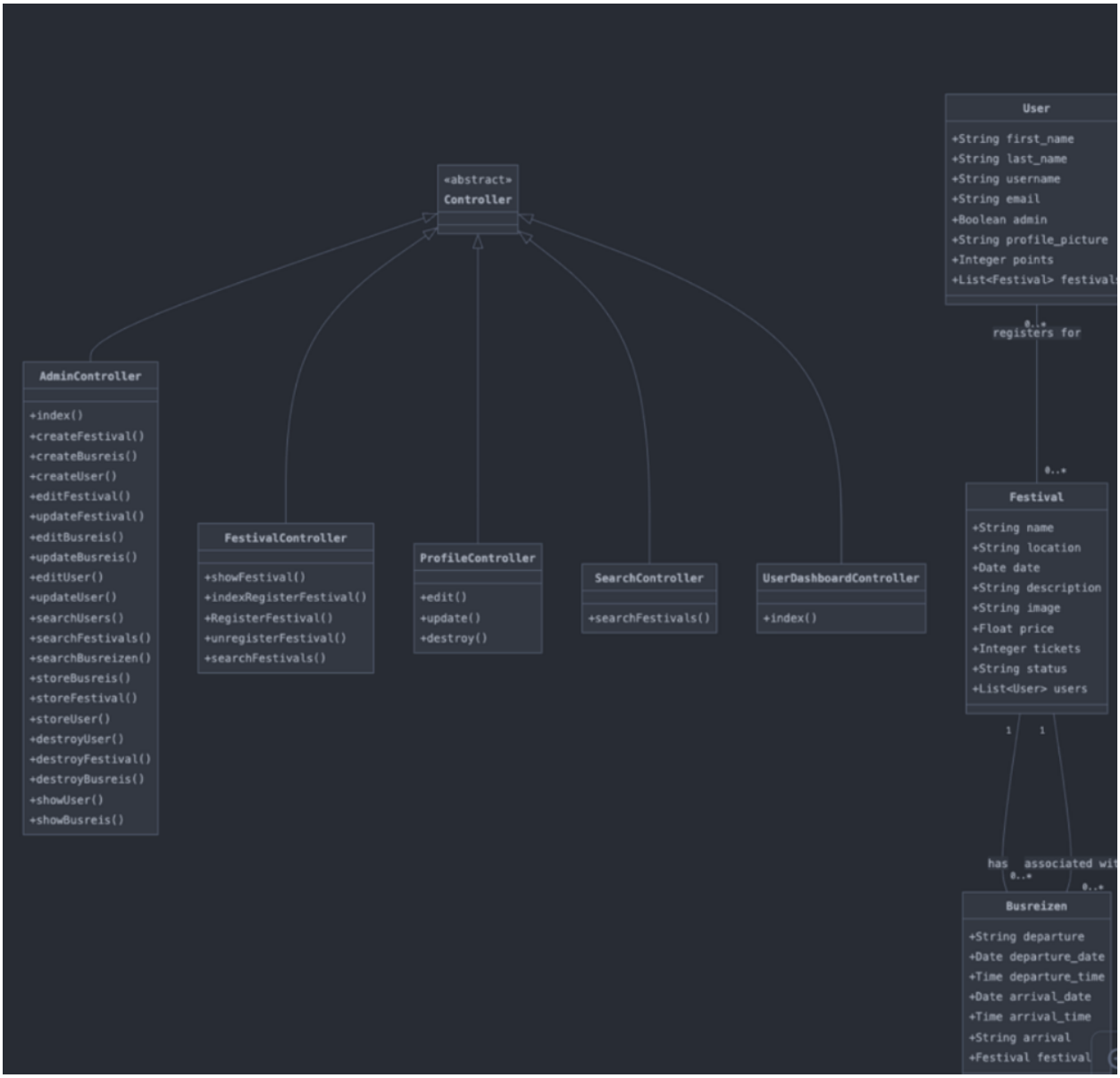
De applicatie is gebaseerd op een klassieke **MVC-architectuur**, waarbij logica, weergave en routing gescheiden zijn voor onderhoudbaarheid. De frontend communiceert met de backend via routes die door Laravel worden afgehandeld. Interactie tussen onderdelen verloopt als volgt:

Routes:



Toelichting:
Dit diagram beschrijft de structuur van een webapplicatie waarin routes worden opgesplitst in publieke, gebruikersspecifieke en admin-only routes. Authenticatie en middleware bepalen de toegang tot gebruikersdashboards, beheerpagina's en algemene functionaliteiten zoals login en registratie.

Controllers:

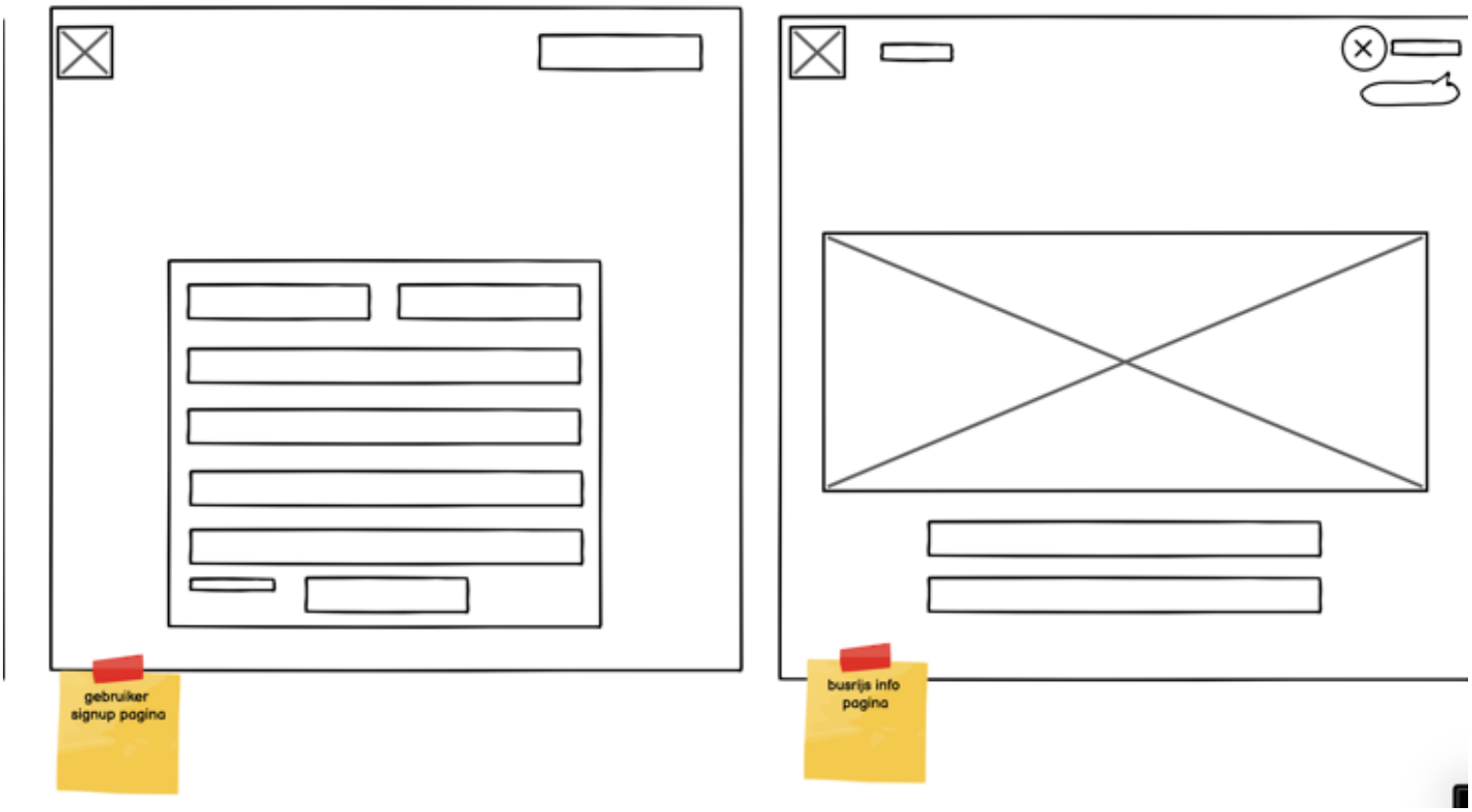


Toelichting:
Dit controllermodel beschrijft de structuur van een MVC-applicatie waarin verschillende controllers, afgeleid van een abstracte Controller, verantwoordelijk zijn voor gebruikersbeheer, festivalregistraties, busreizen en zoekfunctionaliteit. De AdminController beheert alles, terwijl andere controllers gericht zijn op specifieke gebruikersacties zoals profielbeheer, zoeken en festivalregistratie.

4.0 Interface ontwerp

Op basis van de functionele en niet-functionele requirements, zoals beschreven in hoofdstuk 2, en het technische ontwerp uit hoofdstuk 3, zijn de volgende wireframes ontworpen.

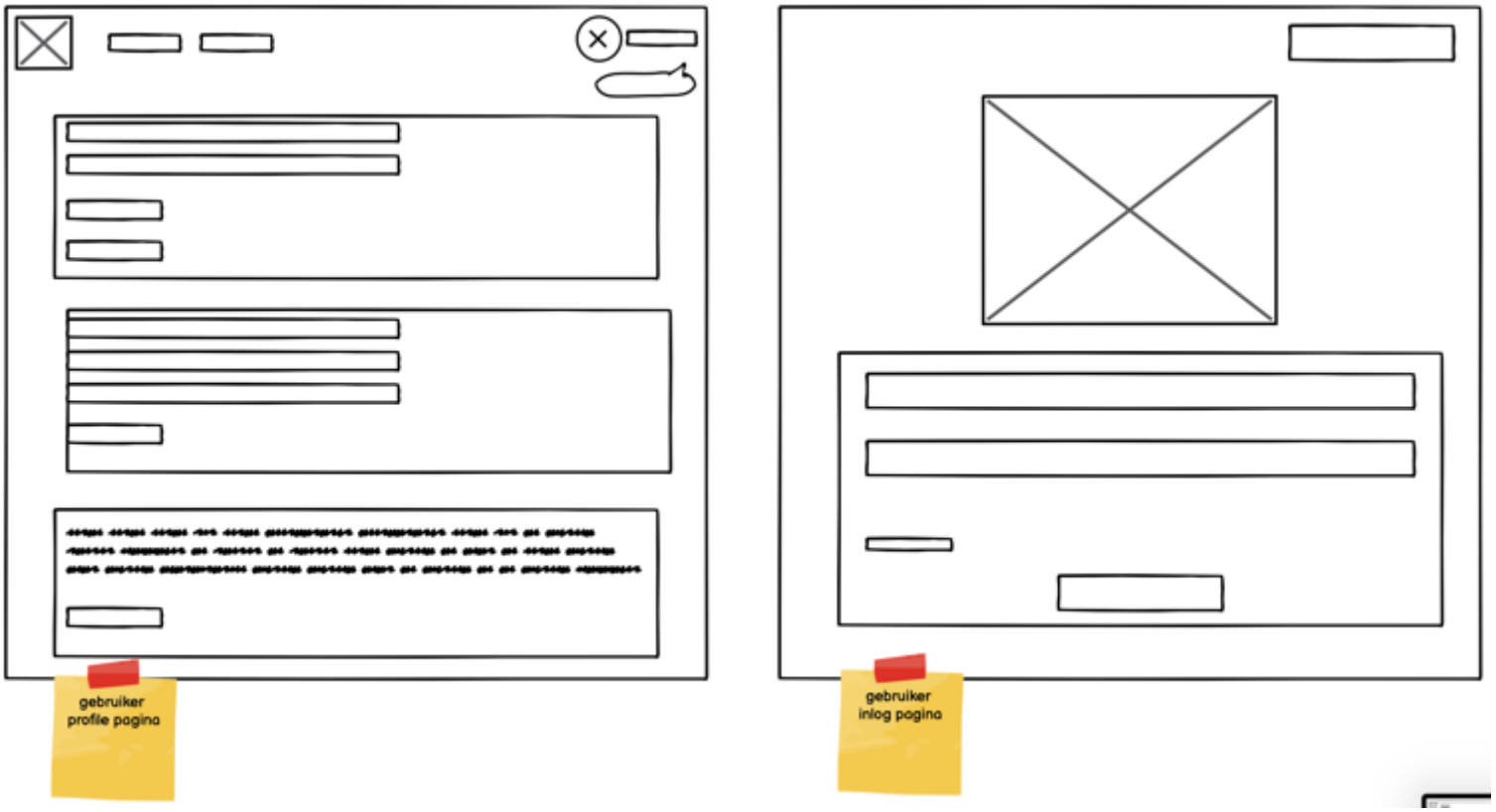
Gebruiker signup / bus reis informatie



Links: Hier kunnen nieuwe gebruikers zich registreren met naam, e-mailadres, wachtwoord en geboortedatum. De invoervelden zijn voorzien van validatiefeedback.

Rechts: Toont gedetailleerde informatie over een specifieke busreis, inclusief opstaplocaties, tijden en beschikbaarheid.

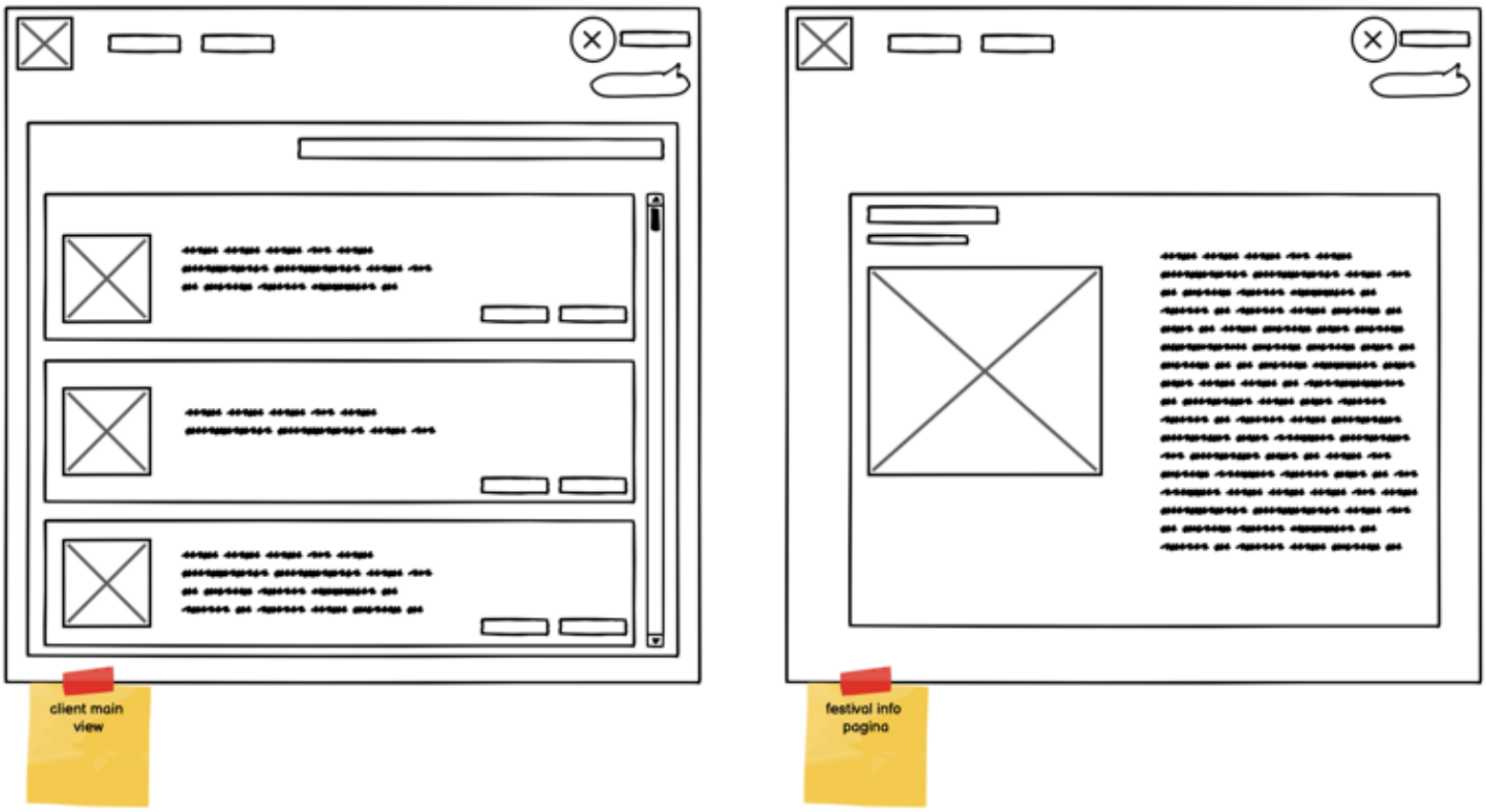
Gebruiker profile pagina / gebruiker inlog pagina



Links: Geeft het persoonlijke dashboard van de gebruiker weer, met profielgegevens, loyaliteitspunten en boekingsgeschiedenis.

Rechts: Toont een eenvoudige inloginterface met velden voor e-mail en wachtwoord. Bij foutieve invoer verschijnt een duidelijke foutmelding.

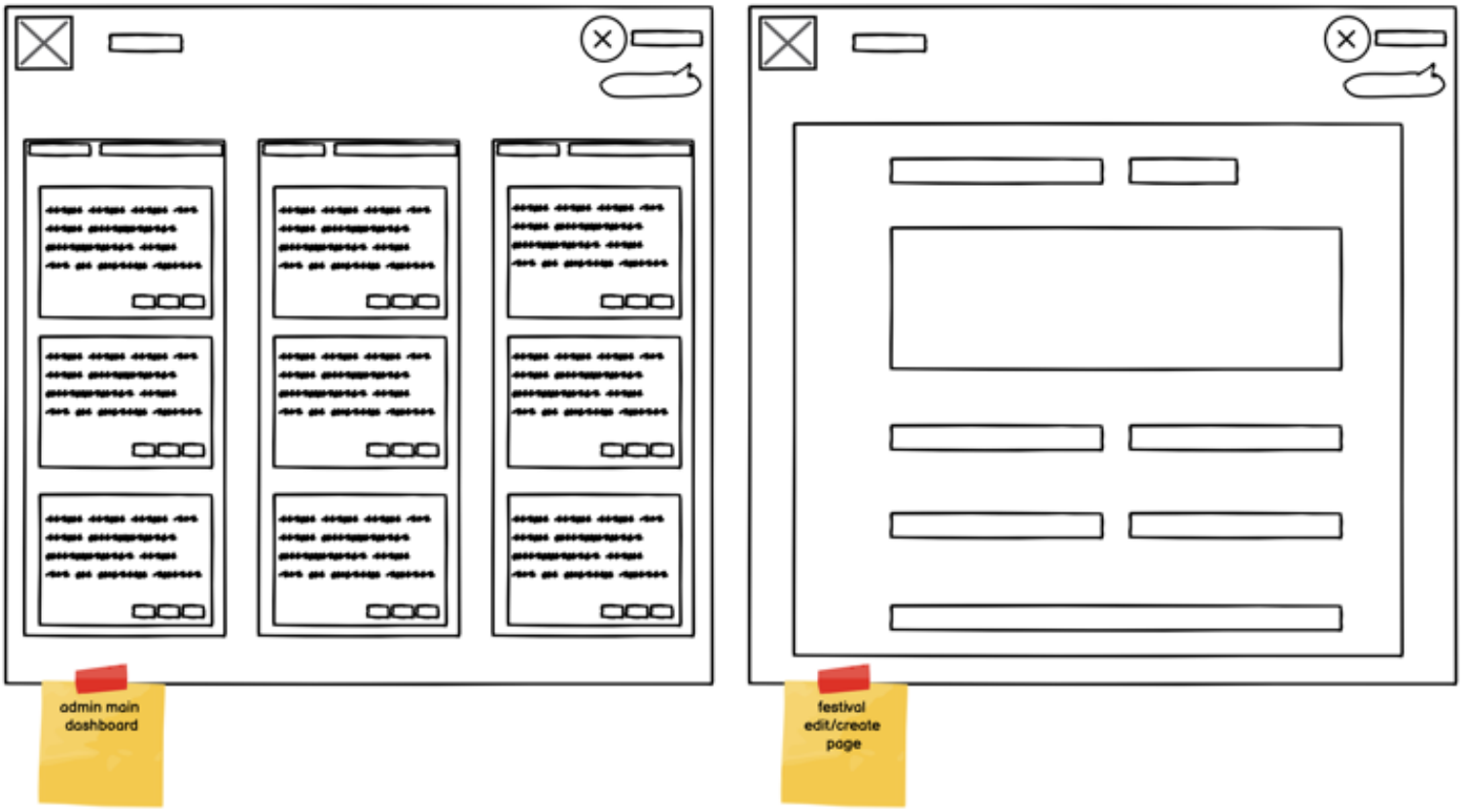
Gebruiker hoofdscherm / festival informatie



Links: De landingspagina na inloggen, met een overzicht van beschikbare festivals en filters op datum, locatie en genre.

Rechts: Geeft uitgebreide informatie over een geselecteerd festival, inclusief beschrijving, locatie, beschikbare data en bijbehorende busreizen.

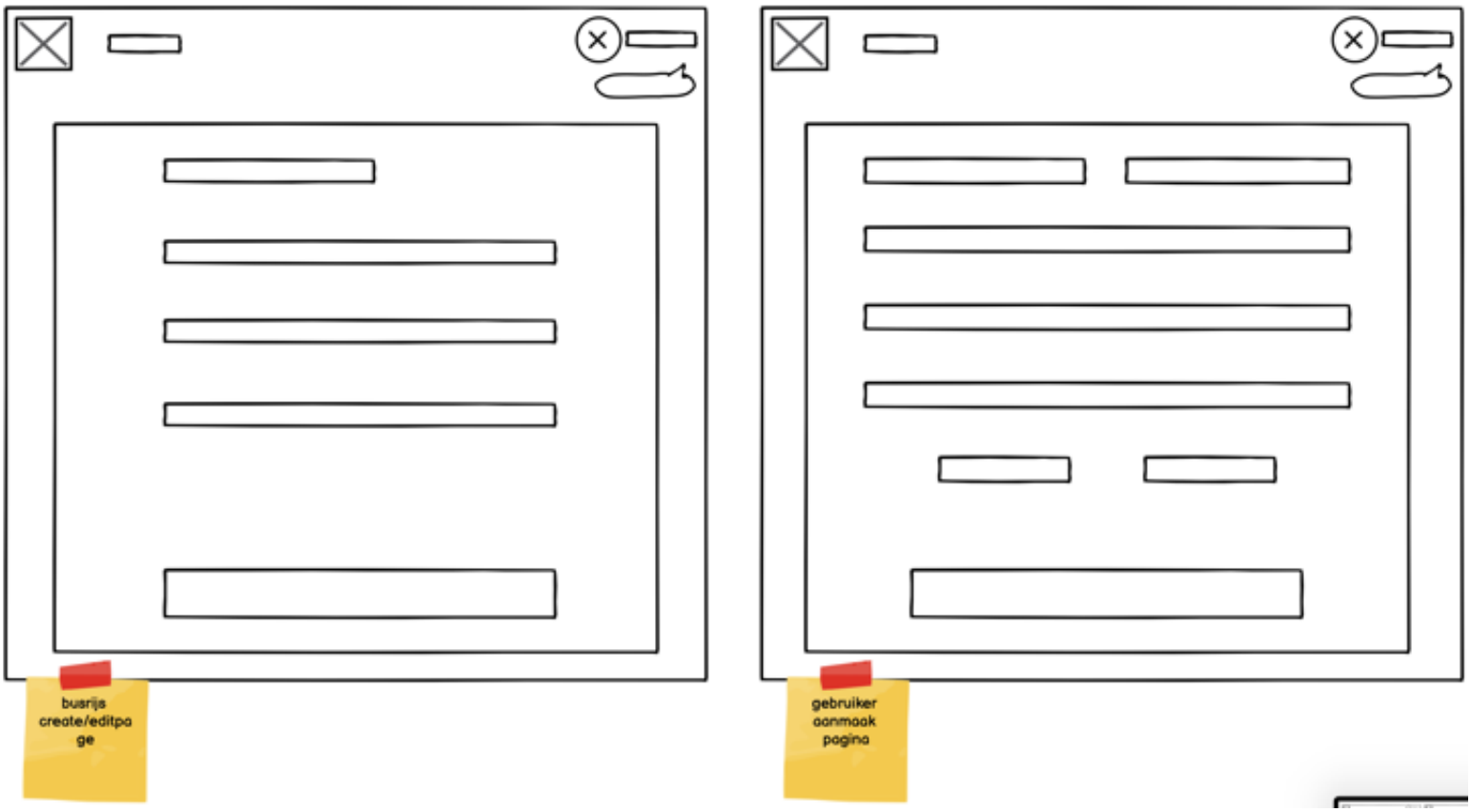
Admin dashboard / festival edit & create



Links: Het overzichtelijke beheerdersportaal waarin alle beheerfunctionaliteiten toegankelijk zijn via een navigatiemenu.

Rechts: Een formulier voor het aanmaken of bewerken van festivalgegevens, inclusief validatie en datumselectie.

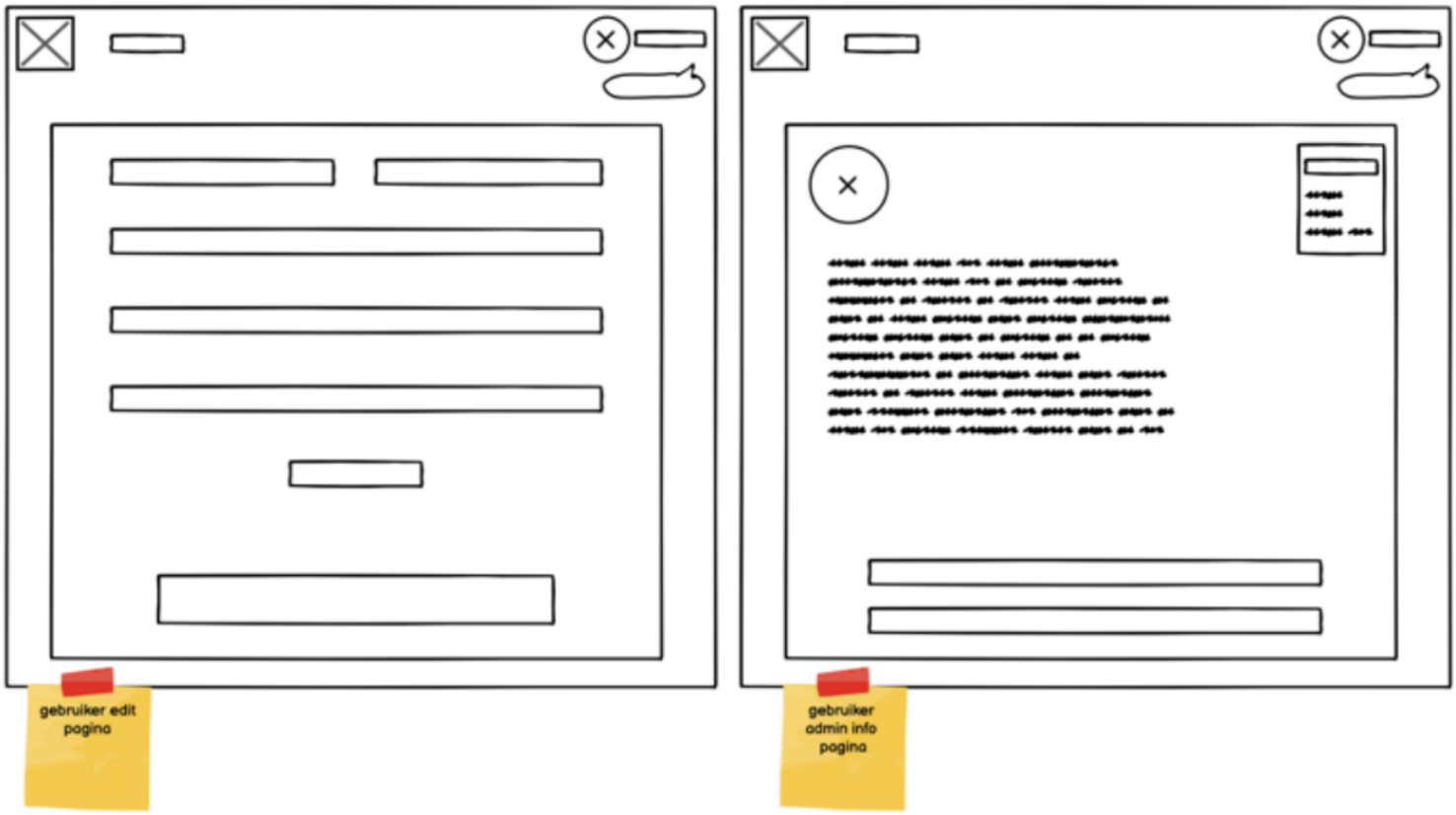
Bus reis edit & create / gebruiker creation



Links: Maakt het mogelijk om busreizen toe te voegen of aan te passen, gekoppeld aan een festival en met details zoals route en capaciteit.

Rechts: Stelt beheerders in staat om nieuwe gebruikers handmatig aan te maken met e-mail, naam, wachtwoord en rechten.

Gebruiker edit / gebruiker admin info



Links: Stelt beheerders in staat om gebruikersgegevens handmatig aan te passen, zoals e-mail, naam of rechten.

Rechts: Geeft een gedetailleerde informatie pagina van de geselecteerde gebruiker met de opgespaarde punten, e-mail, geplande busreizen en festivals ect.

5.0 Testplan

1. Doel van het testplan

Dit testplan beschrijft de teststrategie voor het project, gericht op het valideren van de functionaliteiten zoals beschreven in de requirements. Zowel toegang en permissies als gegevenscreatie worden getest via feature tests.

2. Teststrategie

- Type: **Feature tests (acceptatietesten)** met behulp van Laravel's testing framework.
- Testvorm: **Automatisch** via PHP artisan testing.
- Scope: Authenticatie, autorisatie, en datacreatie (gebruikers, festivals, busreizen).

3. Uitgevoerde Tests

Testclass	Doel	Status
AdminTest	Verifieert toegangscontrole van admin vs user	Geslaagd
CreateDataTest	Verifieert aanmaak van gebruikers, festivals, busreizen	Geslaagd

4. Toelichting per test

- AdminTest
    - **admin can access dashboard** – bevestigt correcte rechten voor admin.
    - **user cant access admin dashboard** – valideert toegangsrestrictie.
- CreateDataTest
    - **user creation** – controleert gebruikersregistratie.
    - **festival creation** – controleert of festivals correct aangemaakt worden.
    - **busreis creation** – controleert succesvolle opslag van busreizen.

6.0 Implementatie & Deployment

6.1 Implementatie

De implementatie van de applicatie is uitgevoerd op basis van het **MVC-principe (Model-View-Controller)**, zoals eerder toegelicht in hoofdstuk 3. Laravel is gebruikt als PHP-framework voor de backendlogica en routing, in combinatie met Tailwind CSS en standaard HTML voor de frontend.

Belangrijke stappen in het implementatieproces:

- **Initialisatie van het project** met Laravel, inclusief configuratie van routes, middleware en authenticatie.
- **Opzetten van de database** met migraties en seeders, op basis van het ERD.
- **Bouwen van controllers** voor gebruikersacties zoals registratie, inloggen, boeken van busreizen en loyaliteitspunten.
- **Ontwikkeling van views** op basis van de ontworpen wireframes, inclusief formulieren, feedbackberichten en navigatie.
- **Implementatie van beheermodules** voor het toevoegen en beheren van festivals, gebruikers en busreizen.
- **Testen en verfijnen** van afzonderlijke onderdelen met behulp van Postman en handmatige gebruikerssimulaties in de browser.

Elke functie werd afzonderlijk ontwikkeld en getest voordat deze in de hoofdbranch werd geïntegreerd. Deze aanpak zorgde voor controleerbare voortgang en beperkte bugs.

6.2 Deployment

Tijdens het project is geen gebruik gemaakt van een externe productieomgeving. De volledige ontwikkeling en uitvoering zijn lokaal gebeurd, passend binnen de scope van een leertraject.

De lokale deploymentomgeving bestond uit:

- **Laravel development server** (php artisan server) voor lokale hosting van de applicatie
- **MySQL database** met lokale toegang via mySQL Workbench
- **Configuratie via .env bestand** voor veilige opslag van databasegegevens en andere omgevingsvariabelen
- **Composer en NPM** voor package- en assetbeheer

Toekomstige uitbreidingsopties voor hosting kunnen zijn:

- **Laravel Forge** of **Ploi** voor automatische serverconfiguratie op een VPS
- **Docker containers** voor draagbaarheid en schaalbaarheid
- **Shared hosting** indien de applicatie licht blijft qua belasting en vereisten