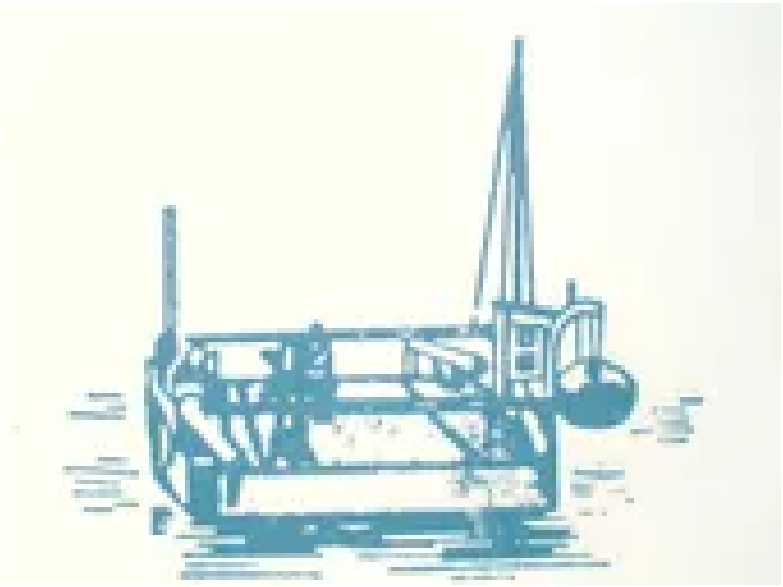
Ontwerpdocument

Ingensche Veer



In opdracht van
University of Applied Sciences Utrecht

Datum: 30 juni 2024
Versie: 2.0
Auteur: Vincent van Setten
Studentnummer: 1734729
Klas: V1C

Inhoudsopgave

1 Revisie Historie

Versie	Datum	Omschrijving
1.0	04-06-2024	Eerste Versie
2.0	30-06-2024	Tweede Versie

Tabel 1: Versiegeschiedenis

2 Inleiding

Tussen Ingen en Elst (Utrecht) ligt een veerpont. Deze gaat eigenlijk constant op en neer, zonder een vast tijdschema. De veerpont is vrij groot en is voor zowel voetgangers, als auto's en fietsen. Er passen ongeveer 20 auto's op en een groot aantal fietsers en voetgangers. De veerpont is relatief druk. Er worden dagelijks honderden mensen overgezet en is erg belangrijk voor de inwoners rondom Ingen en Elst. Het lastige met de drukte op de veerpont is dat het vaak in vlagen komt. De ene overtocht heeft slechts twee auto's en bij de volgende staan er zoveel dat niet alle auto's tegelijk over kunnen. Dit komt vooral omdat mensen niet weten wanneer de veerpont aankomt, waardoor veel mensen hem vaak net missen en vervolgens lang moeten wachten.

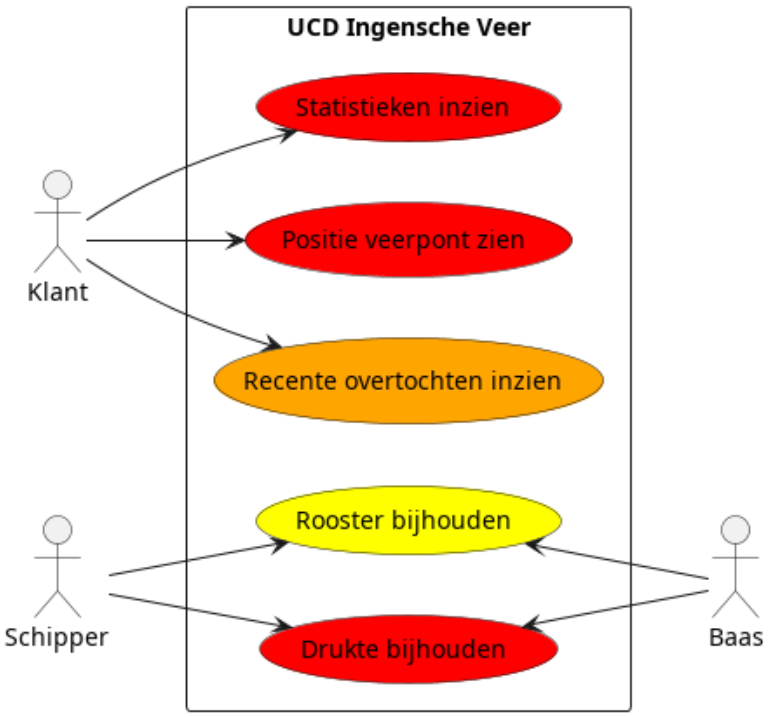
Dat zorgt voor veel frustratie bij de klanten en veroorzaakt ook stress bij de schipper. De kern van het probleem is dat klanten de aankomsttijd van de veerpont niet kunnen inschatten. Een mogelijke oplossing voor dit probleem is de ontwikkeling van een webapplicatie die de huidige locatie van de veerpont weergeeft. Met deze applicatie kunnen klanten inschatten wanneer de veerpont aankomt, wat helpt bij het spreiden van de drukte en het verminderen van de wachttijden. Dit systeem zal gebruik maken van AIS-data om real-time informatie over de veerpont te bieden.

De webapplicatie helpt ook met een ander probleem. Als het erg druk is, kan de baas van de veerpont extra hulp inschakelen. Momenteel gaat dat door puur te kijken op een bepaald moment hoe druk het is. Hiermee komt de hulp vaak net te laat, waardoor er een grote ophoping aan drukte ontstaat. Het is de bedoeling dat met de webapplicatie de schipper per overtocht kan aangeven hoe druk het is. Hiermee kan de baas zien wanneer er hulp nodig is, maar kan hiermee ook inschatten hoe druk het gaat worden op basis van historische data.

In dit ontwerpdocument wordt beschreven welke dingen er worden opgeleverd aan het eind van het IPASS project, hoe het project wordt aangepakt en er uiteindelijk uit gaat zien. Ook worden de risico's en de planning van het project beschreven.

3 Overzicht van het systeem

4 Use Cases



Figuur 1: Use Case Diagram - Ingensche Veer

In het use case diagram in figuur 1 zie je de verschillende use cases. Deze zijn gekleurd naar het MoSCoW model. Rode use cases zijn Must haves, oranje use cases zijn Should haves, gele use cases zijn Could haves en groene use cases zijn Won't haves. Het use case diagram laat alleen de daadwerkelijke geimplementeerde use cases zien. Terugblikkend op het Plan van Aanpak is er vrij veel geimplementeerd. Van de Must Haves is alles geimplementeerd. van de Should Haves is ook alles geimplementeerd. Van de Could Haves zijn de roosters geimplementeerd, maar de geschatte drukte, snelheid van invididuele schippers en integratie met weersomstandigheden niet. Van de Won't Haves is niks geimplementeerd. Alle statistieken van de Must haves, could haves and should haves zijn geimplementeerd onder de enkele use case 'statistieken inzien'.

4.1 Actoren

Actoren	**Omschrijving**
Baas	De baas is de eigenaar van de veerpont.
Klant	De klant is de persoon die gebruik maakt van de veerpont.
Schipper	De schipper is de persoon die de veerpont vaart.

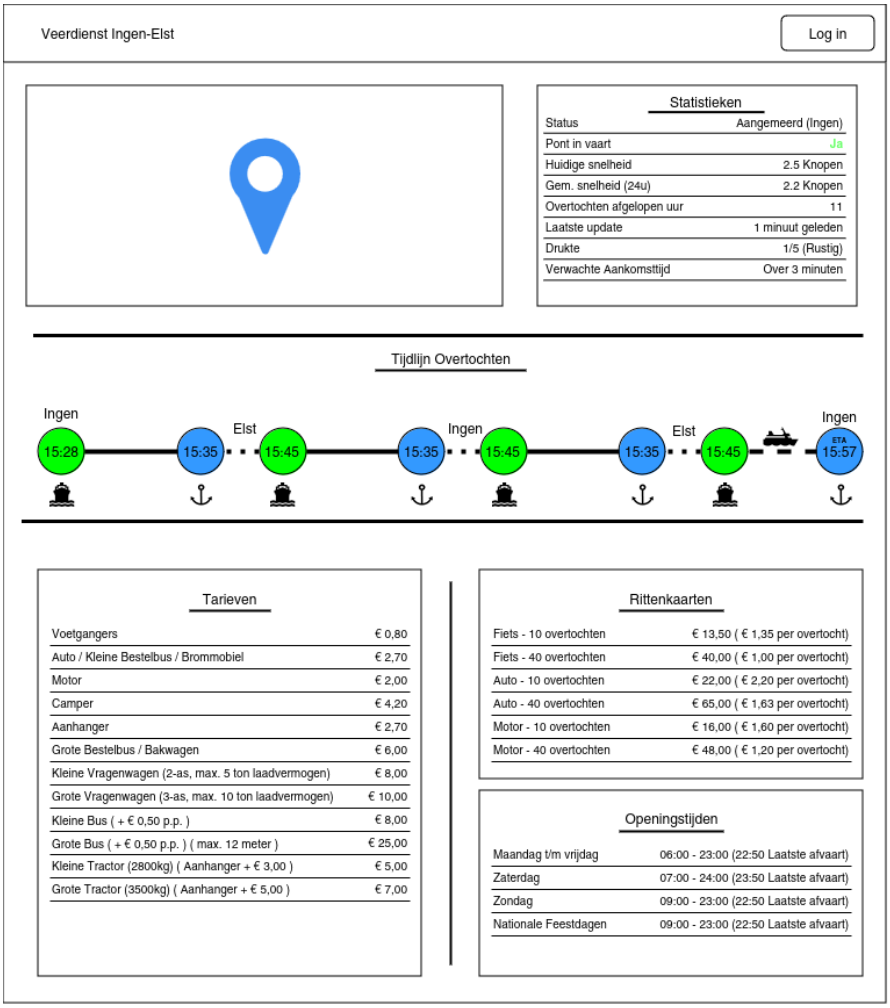
Tabel 2: Actoren

4.2 Use case beschrijvingen en wireframes

4.2.1 UC01 - Bekijk locatie veerpont

ID: UC01	**Use Case Naam:** Bekijk locatie veerpont
Actoren:	Klant
Samenvatting:	De klant kan de actuele positie van de veerpont inzien op een kaart
Preconditities:	De veerpont stuurt AIS signalen en deze worden via de API ontvangen.
Scenario:	1. De klant opent de webapplicatie 2. Systeem haalt de actuele locatie van de veerpont op. 3. Systeem toont de locatie van de veerpont op een kaart. 4. De klant kan de locatie van de veerpont inzien.
Postcondities:	nvt

Tabel 3: Use Case Beschrijving UC01 - Bekijk locatie veerpont



Figuur 2: Wireframe UC01 - Locatie veerpont bekijken

5 Modellen

5.1 Domeinmodel

Figuur 3: Functioneel Domeinmodel - Ingensche Veer

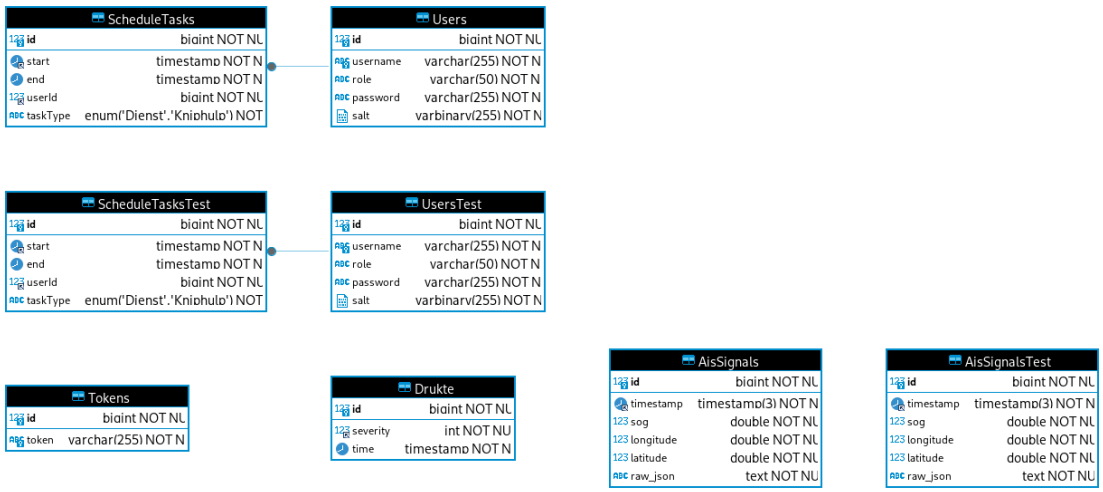
In het domeinmodel in figuur 3 zie je elke klassen in mijn implementatie. Bovenin zijn de verschillende resources te zien. De resource klassen regelen de API calls. Aan deze klassen zijn verschillende andere klassen gekoppeld die ze nodig hebben. Dit zijn veelal data klassen. Zo gebruiken de resources veel 'request' en 'result' klassen, zoals de TaskCreateRequest. Dit is een simpele klasse die helpt met het serialiseren en deserialiseren van de objecten naar de frontend. In het midden van het model zijn een aantal Database controller klassen. De klasse 'DatabaseStorageController' is de storage controller die weer verschillende DatabaseController klassen heeft. Deze klassen maken verbinding met hun eigen database tabel. Zo is DatabaseAisController verantwoordelijk voor de tabel met de AIS signalen in de database. Rechtsonderin het diagram is de AisSignal klassen te zien, samen met de subklassen. Dit zijn simpelweg json implemtatie klassen, om de inkomende json data te kunnen parsen. De testklassen zijn buiten beschouwing gelaten in dit diagram, omdat deze niet relevant zijn voor het domeinmodel en het diagram onnodig nog groter maken.

5.2 Business Rules

Bij het domeinmodel in figuur 3 horen de volgende business rules:

1. **Attribute Rule:** Een veerpont heeft minimaal een schippers.
2. **Attribute Rule:** Een schipper heeft altijd maar één baas.
3. **Entity Rule:** Elk AisSignaal heeft een snelheid, positie en tijd.
4. **Entity Rule:** Elke gebruiker heeft een gebruikersnaam en wachtwoord.
5. **Tuple Rule:** De datum van het verteksignaal is altijd kleiner dan de datum van het aankomstsignaal.

5.3 Datamodellen



Figuur 4: Database ERD - Ingensche Veer

In figuur 4 is het ERD van de database te zien. Sommige tabellen hebben een kopie. Deze heten ¡tabel¿Test. Dit zijn tabellen die gebruikt worden tijdens de tests, zodat deze niet de echte data aantasten. Ik zal de entiteiten hieronder kort toelichten.

1. **ScheduleTasks**: Deze tabel bevat een taak. Een taak is een dienst op de veerpont. Deze is gekoppeld aan een gebruiker.
2. **Users**: Deze tabel bevat de gebruikers en inlogdata van de gebruikers.
3. **Tokens**: Deze tabel bevat de JWT tokens. Deze tabel is eigenlijk onnodig en dit had lokaal gedaan kunnen worden. Ik kwam er pas later achter dat tokens niet hoeven te persisten, omdat deze geinvalidate worden zodra de applicatie opnieuw opstart.
4. **Drukte**: Deze tabel bevat een 'drukte' signaal. Een schipper kan deze doorgeven, wat de baas vervolgens weer kan inzien.
5. **AisSignals**: Deze tabel bevat de AIS data. Dit zijn de signalen die de veerpont uitzendt.

Ik heb geen fysiek of conceptueel datamodel gemaakt, omdat ik dit nooit heb gehad. Ik heb een oude versie van modelling gevolgd(in 2019) en dit niet gehad.

6 Technologieen

1. UML
2. Java
3. HTML
4. CSS
5. Javascript
6. Jax-RS(Rest)
7. HTTP-Protocol
8. MySQL

7 Overdracht

7.1 Installatie

1. Clone de repository
2. Ga naar de frontend folder
3. Run 'npm install'
4. Run 'npm run build'
5. Ga terug naar de source folder
6. Open het project in intellij idea
7. Laad de maven dependencies in de pom.xml
8. Stel de applicatie in als tomcat applicatie
9. Run de applicatie

7.2 Wachtwoorden

- **Vincent**: admin
- **Stephan**: schipper
- **Vincentvl**: schipper
- **Baas**: baas

8 Referenties

Er zijn geen referenties.