

2022-2023

Examenproject – 1^e zittijd

Winkelbeheer



Dit document beschrijft het examenproject voor de **1^e zittijd** van het vak User Interfaces 3. Het project wordt **individueel** uitgewerkt.

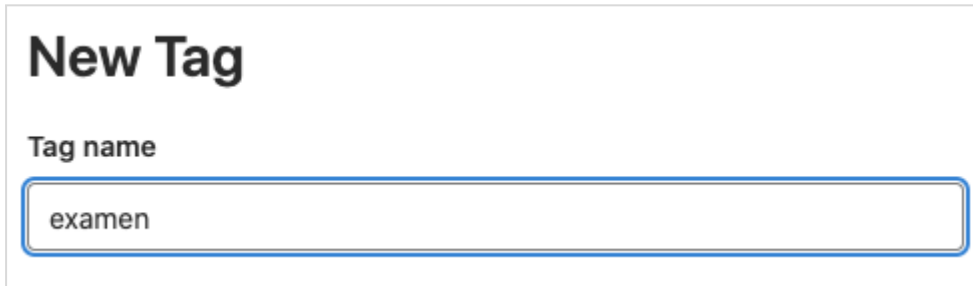
Inhoud

Uitwerking en oplevering	2
Beoordeling	2
Project	3
Vaste functionaliteit	3
Te kiezen functionaliteit (minimaal 1 uit lijst)	5
Wireframes (sketches) en usability testen	6
Opmerkingen	6

Uitwerking en oplevering

Elke student krijgt een repo toegewezen onder de groep 'Kdg-TI - User Interface 3'. Het project wordt hierin opgebouwd in verschillende commits (commit na elke werksessie). Ten laatste op **donderdag 10 november 2022 om 8:00u** wordt het project opgeleverd. Dit gebeurt als volgt

1. Maak een tag 'examen' aan in de gitlab repo



2. Download de repo als zip via deze tag



3. Upload de zip file op Canvas in de opdracht [examen eerste zit](#)

Een project dat niet in de toegewezen gitlab repo wordt gemaakt, geen examen tag bevat en niet via Canvas wordt opgeleverd op de gestelde deadline **wordt niet geëvalueerd**.

Beoordeling

Tijdens het mondeling examen (on campus) demonstreer je het project.

Aan de hand van mondelinge vragen wordt gepeild naar je inzicht in de toegepaste technieken en de concepten aangebracht in de cursus. Het is mogelijk dat je tijdens het examen een stuk functionaliteit aan het project dient toe te voegen.

De beoordelingsmatrix ziet er als volgt uit (vind je ook op Canvas in PDF):

	Onvoldoende	Behaald	Excellent
Functionaliteit	Slechts een beperkte set features zijn geïmplementeerd. De user experience is ontoereikend en niet getest. De ideal state is met een beperkt aantal gegevens uitgewerkt.	De gevraagde features zijn klaar. De user experience is getest en de verkregen feedback is verwerkt. Zowel ideal, empty als error states zijn uitgewerkt	Er zijn extra features opgeleverd. Er is aandacht besteed aan onboarding, micro-interacties, shortcuts voor vaak voorkomende handelingen etc. De applicatie connecteert emotioneel met de gebruiker.
Kwaliteit	Het framework wordt niet correct toegepast (onvoldoende opdeling in componenten, geen routing, geen hooks...). Er wordt gezondigd tegen best practices. Er is geen foutafhandeling.	Het framework wordt correct toegepast (components, routing, hooks,...). De code is overzichtelijk en bevat geen te grote componenten, methods,... Foutafhandeling staat op punt.	Er worden geavanceerde technieken toegepast die niet of slechts beperkt behandeld zijn in het lesmateriaal
Begrip	De toegepaste concepten kunnen niet of slechts beperkt worden toegelicht. De leerstof is onvoldoende begrepen. Code is gebouwd op basis van voorbeelden zonder in detail te begrijpen wat er gebeurt.	De toegepaste concepten kunnen toegelicht worden. De code, het framework en de tools zitten in de vingers. Er is kennis van de concepten die in het lesmateriaal ter omkadering zijn meegegeven (zie overzicht op Canvas)	Concepten die niet of slechts beperkt werden behandeld kunnen toegelicht worden. Er is inzicht in de voor- en nadelen van een bepaalde aanpak tov andere methoden.

Project

Bouw een applicatie voor de organisatie (wat ligt waar, voorraad,...) van een (fysieke) winkel. De toepassing is in principe onafhankelijk van het soort winkel maar je kiest wel een type (kleding, elektronica, grootwarenhuis,...) zodat je visualisatie en testgegevens hierop gericht kunnen zijn.

Een winkel heeft *afdelingen* en binnen elke afdeling zijn er rechthoekige *locaties* waar *producten* kunnen liggen. We werken voor de eenvoud niet met producten die boven elkaar kunnen liggen. Een locatie wordt maar door 1 product ingenomen maar een product kan op meerdere locaties liggen binnen een afdeling. Elke afdeling heeft een *grondplan*, dit is een bovenaanzicht van de product locaties in die afdeling. Rechthoeken kunnen tegen elkaar liggen om zo visueel een winkelrek te vormen. De witruimte tussen de tegen elkaar liggende rechthoeken zijn de gangpaden in de winkel.

In wat volgt beschrijven we eerst de **vaste functionaliteit**, bijkomend kies je **verplicht nog 1 feature** uit de lijst van [te kiezen functionaliteit](#).

Vaste functionaliteit

Er zijn twee soorten gebruikers:

- **Klant.** Een klant hoeft niet aan te loggen maar kan slechts een beperkt stuk van de functionaliteit gebruiken
- **Manager.** Om de overige functionaliteit te gebruiken moet je de rol van manager hebben. Omdat security in React applicaties via tokens verloopt en json-server (waarmee we de backend simuleren, zie verder) dit niet ondersteunt, laten we het aanloggen zelf uit de scope van het project¹. Om de rol van manager aan te nemen gebruik je daarom een test knopje op de interface waarmee je het e-mailadres (manager@ui3store.be) en de manager rol in een context stopt. Op elk scherm wordt dit e-mailadres weergegeven, inclusief een knop om uit te loggen. De rol wordt gewist uit de context bij het uitloggen.

¹ Dit komt wel aan bod in andere vakken zoals Integratieproject 2

Als klant kan ik

- Producten **zoeken** op basis van
 - naam
 - in promo of niet

Als geen zoekterm is ingegeven worden de producten in promotie getoond

- De details van een product bekijken (op een aparte URL)
 - naam (string)
 - omschrijving (string)
 - afbeelding (string (URL))
 - product eigenschappen (lijst van (eigenschap: string, waarde: string))
(dit is een generieke structuur die voor allerlei soorten producteneigenschappen bruikbaar is)
 - in promo (boolean)
 - prijs (getal)
 - # stuks in voorraad (getal)
 - naam afdeling (string)
 - op een **grondplan** (bovenaanzicht) bekijken op welke locatie rechthoek dit product zich binnen de afdeling bevindt (aangegeven met bv. een kleur, border,...). Default wordt de naam van een product in elke rechthoek getoond samen met eventuele andere gegevens (zie verder feature <Instellingen>).
- Een overzicht bekijken van de afdeling en voor elke afdeling het grondplan bekijken (welke producten liggen waar) en doorklikken naar de details van een product

Als manager kan ik

- **Alles wat een klant kan**
- Details van een **product aanpassen**
 - naam (verplicht veld), omschrijving, afbeelding url
 - prijs
 - in promo
 - #stuks in voorraad
 - toevoegen, verwijderen en updaten (**CRUD**) van **product eigenschappen**
 - minimum voorraad (getal) , zie verder
 - afdeling
 - **product aan een locatie toewijzen** op het grondplan van de afdeling. Indien er reeds een ander product op deze locatie lag, heeft dit andere product nu geen locatie meer.
- In een oogopslag zien welke producten niet meer (of bijna niet meer) in **voorraad** zijn via een overzicht van de **afdelingen** waarbij voor elke afdeling is aangegeven of er producten zijn die uit voorraad dreigen te raken. Doorklikken op een afdeling toont het **grondplan** van die afdeling. Hierop wordt met kleur en een getal de voorraad voor elk product aangegeven voor de locatie(s) waarop dit product ligt (je mag dezelfde kleur/getal gebruiken indien het product op meerdere locaties ligt):
 - Rood: niet meer in voorraad
 - Oranje: voorraad is kleiner dan de vereiste minimum voorraad voor het product

- Groen: voorraad is op pijl
 - Wit: locatie is niet ingenomen door een product
- Via het grondplan kan je ook doorklikken naar de product details.

Navigatie in de applicatie gebeurt door middel van een navigation drawer en routing

Te kiezen functionaliteit (minimaal 1 uit lijst)

- Instellingen.** De manager kan een aantal instellingen aanpassen die worden opgeslagen in de backend:
 - o Product naam al dan niet weergeven op het grondplan
 - o Product prijs al dan niet weergeven op het grondplan
 - o Of producten in promo zijn al dan niet weergeven op het grondplan
 - o De kleuren instellen die gebruikt worden om op de plattegrond de voorraad aan te geven (default rood, oranje, groen, wit)

Een gebruiker kan kiezen tussen light of dark mode.
- Geavanceerd product overzicht.** De gebruiker kan producten filteren en sorteren op hun eigenschappen. Het gebruik van [full text search](#) uit json-server is voor deze functies niet toegestaan. Zorg er voor dat er - bovenop de naam en 'in promo' filters uit de basisvereisten - kan gefilterd worden op (een combinatie van):
 - o Een prijs bereik (kleiner dan, groter dan)
 - o Een afdeling (dropdown)
 - o Alle producteigenschappen als strings (zie json-server '_like'). Voorbeelden:
 - Je product heeft een eigenschap 'fabrikant': dan kan je zoeken op (een deel van) de naam van die fabrikant
 - Je product heeft een eigenschap 'gewicht', ook dit is opgeslagen als een string. Je kan dus niet met 'groter dan' en 'kleiner dan' werken zoals bij prijs. Het is met andere woorden aanvaardbaar dat als het gewicht bv "40" is, dit resultaat ook zal verschijnen als je als zoekterm "4" opgeeft.
- Producten** kunnen aanmaken, aanpassen en verwijderen (**CRUD**).
(opm. aanpassen van een product zit al in de standaard functionaliteit, zie hoger)
- Ondersteuning voor **producten** die in een rek **boven elkaar kunnen liggen**. Gebruik je creativiteit om dit elegant weer te geven Het volstaat hierbij niet om het bovenaanzicht in hokjes te verdelen, de gebruiker moet een duidelijk beeld krijgen van welke producten er allemaal in het rek liggen (op welke verdieping binnen het rek) en de manager moet een goed beeld hebben van de voorraad van elk product. Het gewone bovenaanzicht blijft tevens behouden (de voorraad kleur wordt dan bepaald door het product met laagste voorraad binnen het rek).
- Responsive grondplan.** Het grondplan is bruikbaar op alle schermgroottes zonder scrollbars. De aspect ratio dient behouden te blijven bij herschalen.

- f. **Plattegrond editor.** CRUD van locatie rechthoeken en op positie zetten met behulp van drag & drop.
- g. **Authenticatie** mbv een IDP (Identity Provider) voor de rol van store manager
- h. **Node.js backend en database** (te vervanging van json-server, zie verder)
- i. **Gebruik van Next.js:** Bouw je applicatie in Next.js. Maak in dit geval gebruik van server side rendering om je producten te tonen op de website.

Wireframes (sketches) en usability testen

Maak **wireframes** op waaruit blijkt dat je vooraf hebt nagedacht over je interface. Denk 'mobile first'. Foto('s) van wireframes op papier (**sketches**) zijn prima. Plaats ze in een folder op de root van je project.

Voer key-task **usability testen** uit met een (beknopt) verslag op de root van je project:

- o Zoek een bepaald product en bekijk de details en ligging ervan (klant)
- o Voeg een eigenschap toe aan een product (manager)
- o Plaats een product X op een bepaalde positie Y in afdeling Z (manager)

Opmerkingen

- Zie de **beoordelingsmatrix** op Canvas voor criteria rond code quality, UX etc..
- De toepassing werkt correct in **evergreen browsers** (chrome, firefox,...)
- De code wordt geschreven met **React met functional components**.
 - o Gebruik van Typescript is verplicht.
 - o Gebruik van routing is verplicht
 - o Gebruik van een component framework (Material-UI,...) is aangewezen maar niet verplicht. Indien je ervoor opteert geen component framework te gebruiken, gebruik dan wel een CSS framework zoals Tailwind of Bootstrap. Layouting gebeurt met het framework en/of flexbox.
 - o CSS preprocessors zoals SASS zijn toegelaten maar niet verplicht
 - o Gebruik van React query is niet verplicht (in dat geval
 - o State management met bv. Redux is toegelaten maar niet verplicht
- De applicatie is **responsive** en bruikbaar op zowel mobiel, tablet als desktop zonder horizontal scrolling. Een uitzondering hierop vormt het grondplan tenzij je kiest voor feature <Responsive grondplan>.

- Men hoeft **niet aan te loggen** (tenzij je kiest voor feature <Authenticatie>). De rol van store manager kan gezet worden met behulp van een testknopje op de interface. De rol wordt beschikbaar gemaakt aan de component tree via een **React Context**.
- Er mag vanuit gegaan worden dat een product locatie **rechthoekig** van vorm is en dus met een standaard HTML element (bv. div) met border kan weergegeven worden. **De positie (x,y) en afmetingen (width, height) van de locaties voor gebruik op de plattegrond weergave worden mee aangeleverd door de back-end** (zie verder). Dit mag in px coördinaten. Er mag vanuit gegaan worden dat de data 'klopt' in de zin dat locatie rechthoeken niet overlappen en voldoende groot zijn om gegevens te kunnen bevatten. Er wordt absolute positioning gebruikt. (tip: zet de parent die het grondplan bevat op position:relative zodat je coördinaten tov dit element gelden)
- De applicatie werkt met een **back-end** die wordt 'gesimuleerd' met behulp van [JSON server](#) (tenzij je kiest voor feature <Node.js>). De structuur (opdeling, naamgeving etc..) van de data bepaal je zelf. De back-end levert de data aan van 1 winkel. Gebruik voor minstens 1 afdeling zinvolle gegevens zodat je een mooi grondplan krijgt.
Voeg de **data.json file** toe aan je project op **gitlab**. Voeg een npm script en een devDependency toe in package.json zodat de backend kan opgestart worden zonder dat json-server globaal hoeft geïnstalleerd te zijn (zoals toegepast in de voorbeeldcode).
- Het zoeken van producten wordt zoveel mogelijk op de backend afgehandeld (zie json-server docs).

Veel succes!

Arno Soontjens
Bart Vochten