My Travel Planner

Ontwikkel document.

Jorne te Nijenhuis.

Inhoud

[Opdracht. 3](#_Toc71721901)

[Ontwerp. 4](#_Toc71721902)

[Technieken. 5](#_Toc71721903)

[React.js 5](#_Toc71721904)

[Axios. 5](#_Toc71721905)

[Lucidchart. 5](#_Toc71721906)

[Trello. 5](#_Toc71721907)

[Libraries. 5](#_Toc71721908)

[Requirements 5](#_Toc71721909)

[Must haves 5](#_Toc71721910)

[Should haves 6](#_Toc71721911)

[Could haves 6](#_Toc71721912)

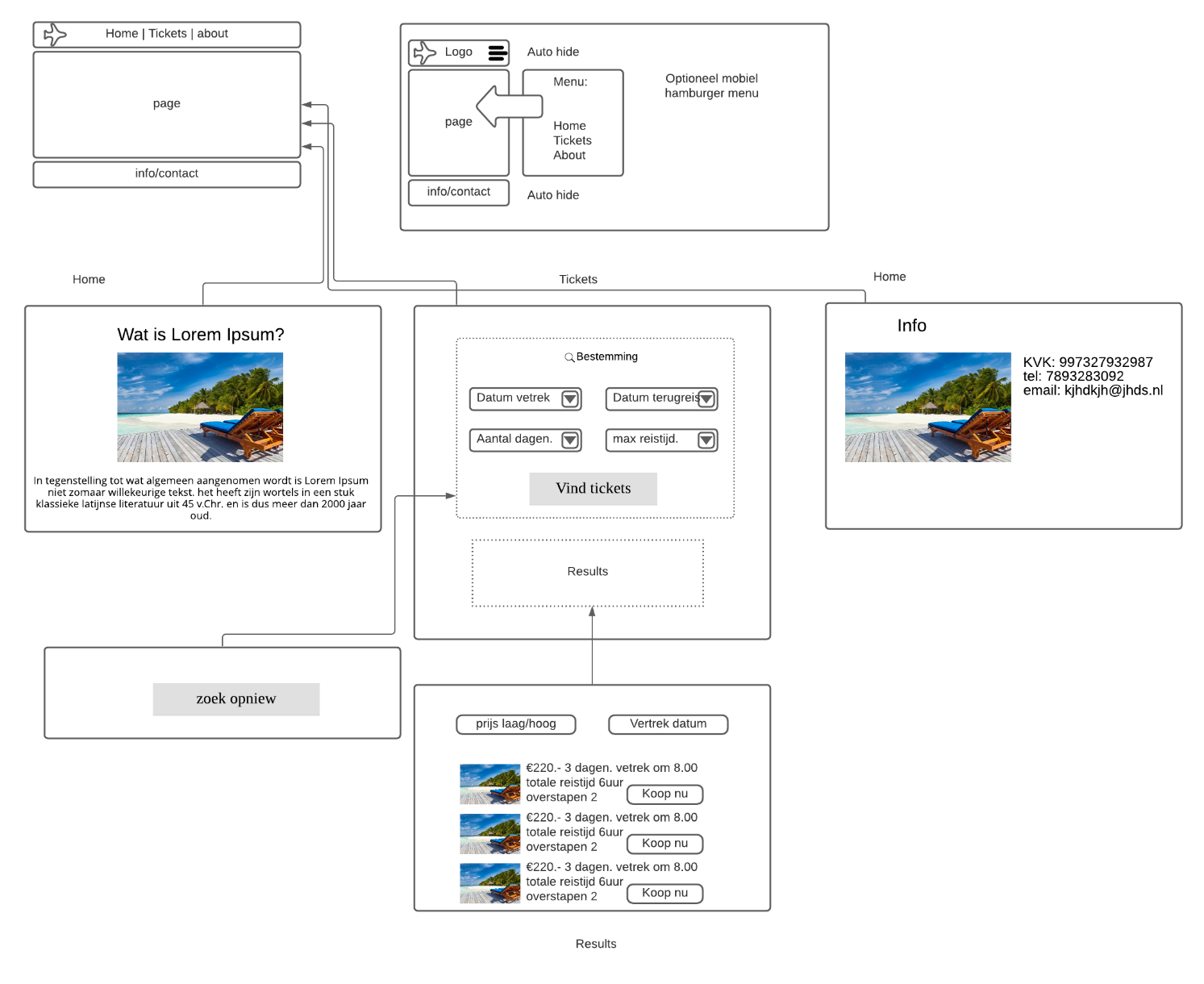
[Won’t haves 6](#_Toc71721913)

[Ontwerp keuzes. 6](#_Toc71721914)

# Opdracht.

In opdracht van Mobiel.nl heb ik een reisplanner gemaakt met gebruik van React.js en de Transavia API. De reisplanner is functioneel maar basic, wel is de code zo opgezet dat er makkelijk uitgebreid kan worden.  
Alle *Must haves* zijn geïmplementeerd en ook een aantal *Should en Could haves.* In dit document zal ik ingaan op het *ontwerp*, *technieken*, *requirements* en *ontwerpkeuzes* van het project.

# Ontwerp.



Ik heb ervoor gekozen om een simpele webapp te ontwerpen. Het ontwerp bleek redelijk te voldoen. Er zijn slechts een paar cosmetische componenten komen te vervallen, zoals het mobiele menu en de afbeeldingen naast de gevonden tickets. Ook is op moment van dit schrijven de Homepagina en info pagina nog niet opgevuld.

Waar ik tegen aan liep was dat de tickets pagina beter had kunnen bestaan uit de objecten: Zoek functie en resultaten. Nu heb ik op een niet zo’n mooie manier het zoek gedeelte laten verbergen als er een zoekactie wordt gedaan. Ook bevat dit component nu te veel logica.

Hier is een link naar het ontwerp:

<https://lucid.app/lucidchart/invitations/accept/inv_741cda01-04eb-4979-824e-d05d0a1987e0?viewport_loc=565%2C-300%2C1186%2C1813%2C0_0>

# Technieken.

## React.js

React is een open-source, front-end, Javascript library. Hiermee kun je user interfaces en UI componenten bouwen. Een groot voordeel van React en vergelijkbare libraries (zoals Vue.js), is dat je slechts één keer de applicatie hoeft op te vragen van de server. Dit is de eerste keer dat ik met React heb gewerkt.  
  
Transavia Api.  
De Api van Transavia is erg gebruiksvriendelijk en heeft duidelijke documentatie. Ook is de service afdeling erg responsieve. Nadat ik een aantal dagen heb gewacht om toegang te krijgen tot de Api-services heb ik een mailtje gestuurd en vrijwel direct werd mij toegang verleend tot de Api. Dit was de eerste keer dat ik connectie heb gelegd met een Api.

Axios.  
Axios is een promise based http-client voor Node.js. Met Axios kun je connectie leggen met Api’s, Back-end en lokale servers. Ik heb gekozen om Axios te gebruiken omdat ik hier al ervaring mee heb.

## Lucidchart.

Om een ontwerp te maken heb ik gebruik gemaakt van de moddeling tool LucidChart. Dit is een online modeleer applicatie waarbij je gemakkelijk kan samenwerken met team genoten om bijvoorbeeld Low-fidelity wireframes of diagrammen te maken.

## Trello.

Om de planning bij te houden heb ik gebruik gemaakt van trello. Hier heb ik net zoals in een scrum planning taken op de Backlog gezet, deze geprioriteerd en bijgehouden.  
dit is een link naar het My Travel Planner-Trelloboard. <https://trello.com/invite/b/0s5dkI9K/eea1a6139852c855f25f01886a15b404/my-travel-planner>

## Libraries.

React-DatePicker, dit levert een makkelijk te implementeren datum selectie component.

React-Select, dit levert een makkelijk te implementeren drop down selectie menu.

# Requirements

## Must haves

* Als gebruiker wil ik, vertekluchthaven kunnen opgeven.
* Als gebruiker wil ik, een minimale reisduur van de vluchten opgeven.
* Als gebruiker wil ik, een maximale reisduur van de vluchten opgeven.
* Als gebruiker wil ik, de periode waarin de vluchten plaatsvinden opgeven.
* Als gebruiker wil ik, het aantal dagen op bestemming opgeven.
* Als gebruiker wil ik, de vluchten kunnen sorteren op prijs, hoog en laag.
* Als gebruiker wil ik, de vluchten kunnen sorteren op Datum.
* Als opdrachtgever wil ik, dat de applicatie gebruik maakt van React-framework.
* Als opdrachtgever wil ik, dat de applicatie gebruik maakt van Hooks.
* Als opdrachtgever wil ik, dat de applicatie gebruik maakt van Styled-components
* Als opdrachtgever wil ik, dat de applicatie gebruik maakt van de Transavia Api.

## Should haves

* Als gebruiker wil ik, een hamburger menu om per pagina te kunnen navigeren.
* Als gebruiker wil ik, een Homepagina.
* Als gebruiker wil ik, een About pagina.
* Als gebruiker wil ik, opnieuw kunnen zoeken om nieuwe resultaten te vinden.
* Als ontwikkelaar wil ik, dat de applicatie unit testen bevat.
* Als opdrachtgever wil ik, dat er gebruikt wordt gemaakt van clean code.

## Could haves

* Als gebruiker wil ik, een melding als mijn ingevulde waarden niet kloppen.
* Als gebruiker wil ik, een alternatieve weergaven op een mobiel device.
* Als ontwikkelaar wil ik, dat de applicatie unit testen bevat.
* Als ontwikkelaar wil ik, een ontwikkel document mb.t. tot dit project.

## Won’t haves

* Als gebruiker wil ik, een werkende Buy-button.
* Als ontwikkelaar wil ik, in het ontwikkel document Apa-bronvermelding.

# Ontwerp keuzes.

## Result of ResultList-component

De afweging om in Tickets.js een lijst met Result-componenten weer te geven of om een component met een ResultList-component. Een Result zou slechts een ticket weergeven en een ResultList zou meerdere tickets kunnen bevatten.  
Ik heb nagedacht over scaling, zowel een Result als een ResultList component kunnen één ticket weergeven en kunnen dus zoals gewenst toegepast worden. Het voordeel van een ResultList is dat je slechts één keer props mee geeft en de *loop* binnen het object laat afspelen. Dit verbetert in mijn optiek de leesbaarheid van de code.

## TicketsList.js sortArray ()

In de sortArray methode wordt zodra er om een date-sortering wordt gevraagd, een copy gemaakt van de originele string die van de Api af komt. De originele string is op de juiste manier gesorteerd en een copy maken van een string lijkt mij minder intensief dan door een Array lopen en hem als reversed terug te geven.

Ook geeft de toevoeging van de resultsCopy[] de optie om altijd terug te gaan naar het originele zoek resultaat.

## Tickets.js

De opzet van Tickets.js bleek helaas niet de juiste te zijn. In het vervolg zal ik mijn objecten verder moeten splitsen. Zo had tickets.js niet de logica moeten bevatten om gebruiker keuzes op te slaan en had dit component ook niet verantwoordelijk moeten zijn voor het connectie leggen met de API om de vliegvelden op te halen.