PARKSPOT 3.0

By Kevin, Jan & Nawang

TASK

Improve traffic flows in city centers by providing better parking advice to drivers.

HOW

Design and develop an app with an amazing user experience

that gives the user a parking spot exactly to his liking.

-

To achieve this we need an extensive database of parking spots and

know whether or not they’re available,

we try to use as much open data as we can and

use computer vision to fill in the blanks.

PLANNING

Deadlines ParkSpot

Datum	Wat	Onderdelen

07/04	Open data bekijken	• Welke data gebruiken we?
		• Welke informatie willen we geven?
		  ○ Prijs/uur?
		  ○ Hoe ver stappen naar bestemming?
		  ○ Max. aantal plaatsen?
		  ○ Grootte voertuig?
		  ○ Rolstoel?
		  ○ Opladen elektrische wagens?
		  ○ ...
		• Specifieke plaats of kansberekening?

14/04	Interviews	• Wat is het belangrijkste voor de gebruiker? • Onderzoeken wat nodig is

21/04	Design	• Wireframes
		• Style Tile
		  ○ Fonts, kleuren, titles, buttons...
		  ○ Consistentie doorheen design
		• Visual Design

28/04	Interviews	• Feedback op design
05/05	Visual Design	• Afgewerkt design
	Coding?	• *Nog niet zeker of we iets moeten opleveren voor Amsterdam* • Welke onderdelen moeten gecodeerd worden?
13/05	Amsterdam	• Presentatie (voornamelijk Lennert, wel hulp nodig bij uitleg werking van de app)
02/06	Coding/App	• Afleveren project ○ *Zodra de deliverables bekend zijn, komen die hier te staan*

USER CASES

0.

Before we start, we need user scenarios.

-

These scenarios are needed to understand your users.

We’ll get our main features from these.

USER CASES

0.

1. Ik wil naar Gent maar ik wil op de Park & Ride staan. Naar Gent wil ik gaan met behulp van mijn persoonlijke geprefereerde navigatie. Ik wil de dichtstbijliggende Park & Ride van mijn eindpositie. De Park & Ride mag niet vol zijn. Vanuit de Park & Ride wil ik genavigeerd worden naar mijn eindbestemming.

2. Ik wil naar Gent maar ik wil liefst zo dicht mogelijk bij mijn eindpositie staan zodat ik niet te veel moet stappen. Ik wil hiervoor mijn geprefereerde navigatiesysteem gebruiken.

3. Ik wil naar Gent maar ik wil ergens staan waar het goedkoop is. Dan wil ik van daar met het openbaar vervoer naar mijn eindbestemming. Ik wil hiervoor mijn geprefereerde navigatiesysteem gebruiken. Ik wil gemakkelijk kunnen betalen. Ik wil van mijn parkeerplaats de juiste dichts bij zijnde tram of bus nemen.

4. Ik wil naar Gent en ik sta het liefst op de ondergrondse parking. Dan wil ik genavigeerd worden te voet naar mijn eindbestemming. Ik wil hiervoor mijn geprefereerde navigatiesysteem gebruiken.

SITEMAP APP

1.

SITEMAP BACKOFFICE

WIREFRAMES

1.

We start by making sketches of the app with

the most essential features.

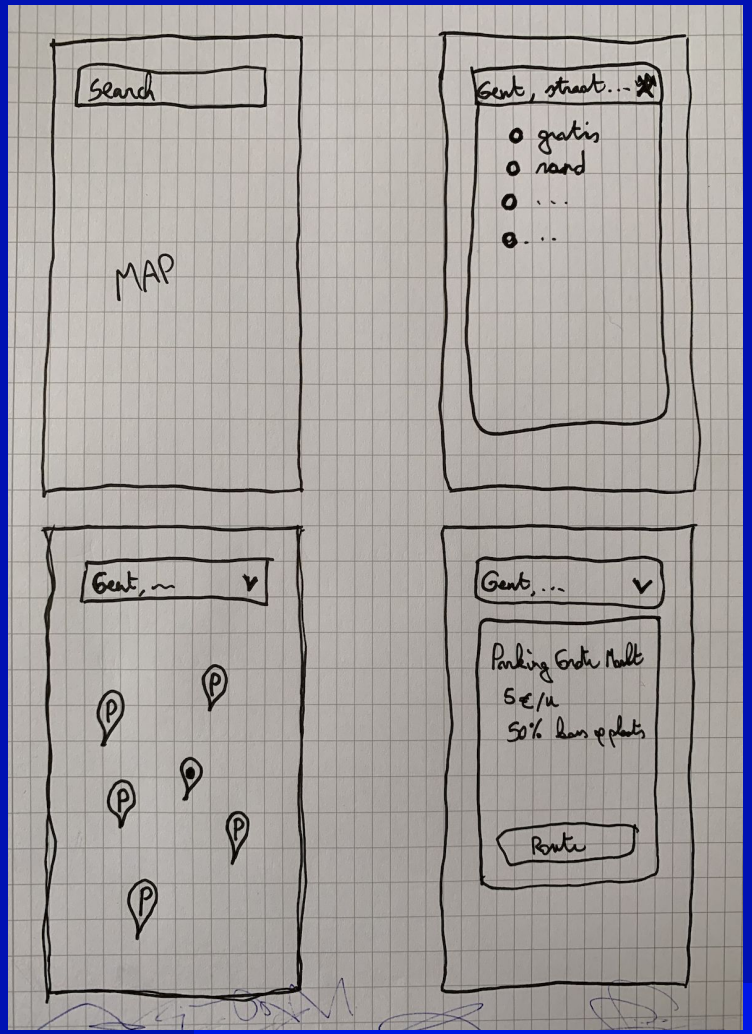
-

Even with those early sketches we proceed to take interviews,

iteration is key

WIREFRAMES

1.



INTERVIEWS

2.

We went to go interview potential users with our wireframes.

-

We got a couple of suggestions like

favorites, history and notifications.

VISUAL DESIGNS

3.

With the interviews in mind we went on to make visual designs.

-

Using Adobe XD we translated our wireframes into something that looks like an actual app.

-

With Adobe XD designs we can give users something that looks like the real thing and test more thoroughly.

VISUAL DESIGNS

3.

INTERVIEWS

4.

With These visuals designs we went to test it on users again.

-

Overall we had positive feedback and

people who couldn't wait for the app to be released.

-

From this point we went on and fine tuned the design.

FINE TUNE

5.

Now it was time to fine tune the designs.

-

We cleaned up the graphics and expanded it to notifications on the main route app and lock screen.

5. FINE TUNE


9:41
Wondelgem
Ghent
Destination...
9:41
Onderbergen 42, Gent
Preferences
Zone
City
Price/hour
€5,00
Distance from destination
<100m
Bancontact
Avoid
Low Emission Zone
Underground
Search
9:41
Parking Savaanstraat (P4)
Savaanstraat 13, 9000 Gent
Price
€2,50/started hour
Type
Underground
Open
24/7
Open spot
87% chance
Onderbergen 42, Gent
Telenet
11:59
Woensdag 8 mei
SPARK
Now
Almost time!
Your time is running out.
Extend your time

FINE TUNE

5.


9:41
13°
Wondelgem
N458
N424
MUIDE
BLOEMEKENSWIJK
R40a
RABOT
R40
Sint-
Amandsberg
PATERSHOL
GHUIZEKEN
MALEM
Ghent
HEIRNIS
Gentbrugg
RIJSENBERGWIJK
Ledeberg
MOSCOU
N43
E17
NIEUW
GENT / UZ
MAALTE
R4
E40
N444
Destination...
Zwijnaarde

CODE

6.

```
/**
 * @function _postDataToAsyncStorage
 * @param {String} key
 * @param {String} value
 * Posts a key with a certain value to the AsyuncStorage
 */
async _postDataToAsyncStorage (key, value) {
    try {
        await AsyncStorage.setItem(key, value);
    } catch (error) {
        console.error(error)
    }
}

/**
 * @function _retrieveDataFromAsyncStorage
 * @param {String} key
 * Will check if there is a key in the AsyncStorage
 */
async _retrieveDataFromAsyncStorage(key) {
    try {
      const value = await AsyncStorage.getItem(key);
      if (value !== null) {
        // We have data!!
        console.log(value);
      } else {
          console.log('no value')
      }
    } catch (error) {
      // Error retrieving data
      console.error(error)
    }
  }
```

CODE

6.

```
/**
 * @function postUserToAuthentication
 * Asynchronous function to post the user to the authentication in the database
 * If the post request was succesfull store the value in the AsyncStorage with the response token from the API
 */
async postUserToAuthentication(){
    const url = "http://192.168.1.5:8080/api/v1/login/local"
    var data = {
        email: this.state.email,
        password: this.state.password
    }

    await fetch(url, {
      method: 'POST', // or 'PUT'
      body: JSON.stringify(data), // data can be `string` or {object}!
      headers:{
        'Content-Type': 'application/json'
      }
    })
    .then(res => res.json())
    .then(response => {
        console.log('Success:', JSON.stringify(response))
        console.log('Token:', response.token)
        this._postDataToAsyncStorage('userToken', response.token)
    })
    .catch(error => {
        console.log('Error:', error)
        this.setState({errorPassVisible: true})

    })
}
```

CODE

6.

```
/**
 * @function render
 * @returns The routing of the app used in the root component      You, 42 minutes ago • adde
 * If a user is not logged in, show different routing.
 */
render() {
    //Check if a user is logged in or not
    if (this.state.loggedIn){
        return(
            <Router>
                <Scene key="root" hideNavBar>
                    <Scene key="home" component={HomeScreen} title="Home" initial={true} />
                </Scene>
            </Router>
        )
    } else {
        return(
            <Router>
                <Scene key="root" hideNavBar>
                    <Scene key="login" component={LogInScreen} title="Login" initial={true}/>
                    <Scene key="register" component={RegisterScreen} title="Register"/>
                    <Scene key="home" component={HomeScreen} title="Home" />
                </Scene>
            </Router>
        )
    }
}
```

CODE

6.

```
/**
 * @function validate
 * @param {String} input
 * @param {String} type
 * Validates the input by a certain type
 * @returns {boolean}       You, an hour ago • added comments
 */
export default function validate(input, type) {
    switch(type) {
        case "email":
            const regexEmail = /^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)*)|(".
            return regexEmail.test(input)
        case "password":
            const regexPass = new RegExp("^(?=.*[a-z])(?=.*[A-Z])(?=.*[0-9])(?=.{8,})");
            return regexPass.test(input)
    }
}
```

CODE

6.

```
getResultsFromAPI = async(zone, price, distance, bancontact, lez, underground) => {
    const url = "http://192.168.5.136:8080/api/v1/searchparkingspots"

    var data = {
        "destinationGeo": {
            "long": this.state.destination.lng,
            "lat": this.state.destination.lat
            },
        "userId": this.state.userId,
        "settings": {
            "zonename": zone,
            "price_per_hour" : price,
            "distance_from_destination" : distance,
            "bankcontact" : bancontact,
            "low_emission_zone" : lez,
            "underground" : underground
            }
    }

    await fetch(url, {
      method: 'POST', // or 'PUT'
      body: JSON.stringify(data), // data can be `string` or {object}!
      headers:{
        'Content-Type': 'application/json'
      }
    })
    .then(res => res.json())
    .then(async(response) => await this.setState({
        parkings: response
    }))
    .catch(error => console.error('Error:', error))
}
```

CODE

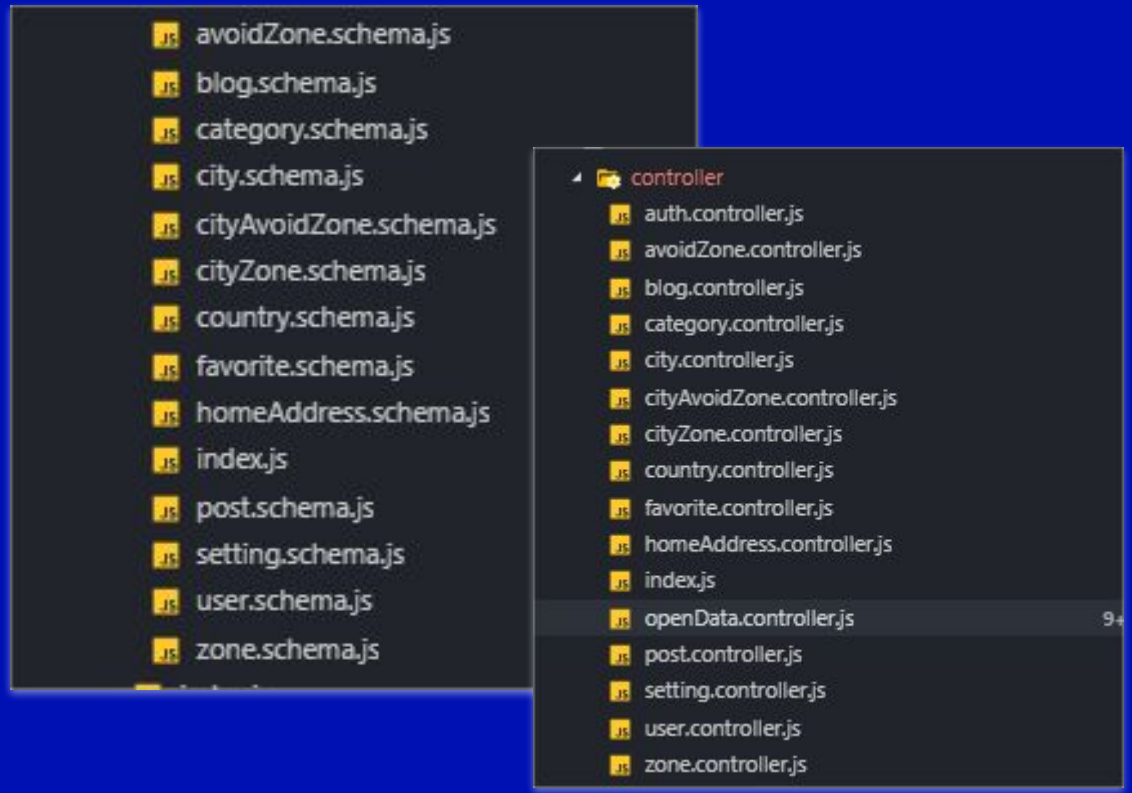
6.

```
render() {
    return (
        <View style={styles.container}>
            <LogOutButton />
            <Map  ref={this.mapElement}/>
            <WeatherIcon/>
            <KeyboardAvoidingView style={styles.cardContainer} behavior="padding" enabled keyboardVerticalOffset={70}>
                <ScrollView ref={(scrollView) => { this.scrollView = scrollView; }} horizontal={true} decelerationRate={0} snapToInterval={width - 20}
                contentInset={{
                    top: 0,
                    left: 10,
                    bottom: 0,
                    right: 10,
                }} style={{display: this.state.cardDisplay}}>

                    {Array.from(this.state.parkings).map(parking => (
                        <Card
                        key={parking.address}
                        onClick={this.showMarkers}
                        destination={this.state.destinationAddress}
                        parkingName={parking.name}
                        address={`${parking.address.match(/(^.*?[^;][0-9])/g)},${(parking.address.split(",")[1])}`}
                        price={parking.price.day}
                        type={parking.type}
                        openWhen={parking.open}
                        chance={parking.chance}
                        />
                    ))}
```

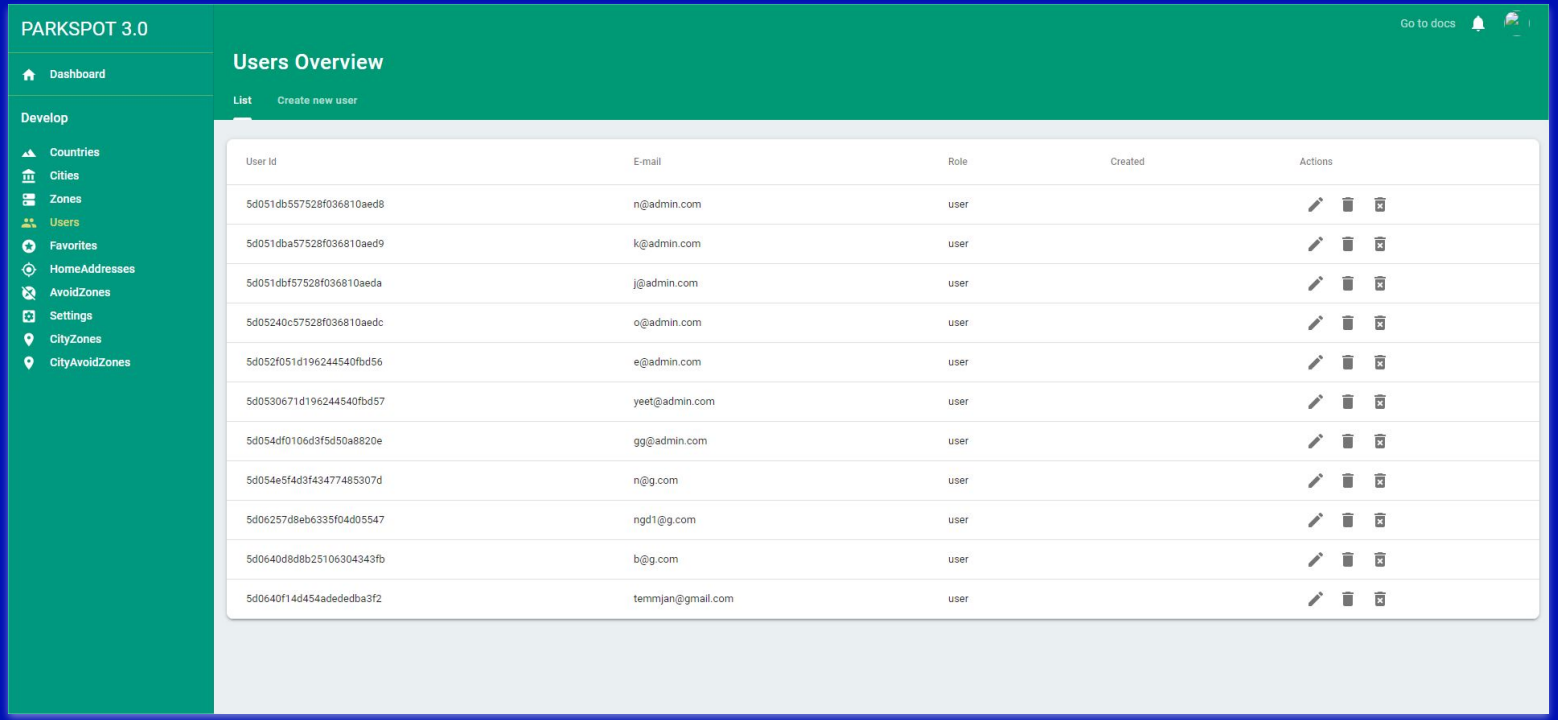
MODELS

7.


avoidZone.schema.js
blog.schema.js
category.schema.js
city.schema.js
cityAvoidZone.schema.js
cityZone.schema.js
country.schema.js
favorite.schema.js
homeAddress.schema.js
index.js
post.schema.js
setting.schema.js
user.schema.js
zone.schema.js
controller
auth.controller.js
avoidZone.controller.js
blog.controller.js
category.controller.js
city.controller.js
cityAvoidZone.controller.js
cityZone.controller.js
country.controller.js
favorite.controller.js
homeAddress.controller.js
index.js
openData.controller.js
post.controller.js
setting.controller.js
user.controller.js
zone.controller.js

Backoffice - UI

8.



DEMO TIMEEE


LIVE DEMO
I ALSO LIKE TO LIVE DANGEROUSLY
memegenerator.net

WE DID IT