Czech Technical University in Prague

Faculty of Electrical Engineering

Department of Radio Engineering

Web Application for Visualization of Air Pollution in a Big Cities

Bachelor's Thesis

Kseniia Chumachenko

Field of study: Communications, multimedia and electronics

Supervisor: Ing. Stanislav Vítek, Ph.D.

May 2019

CTU CZECH TECHNICAL UNIVERSITY IN PRAGUE

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Chumachenko Kseniia**	Personal ID number:	**446427**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Radioelectronics**		
Study program:	**Communications, Multimedia, Electronics**		
Branch of study:	**Multimedia Technology**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Web Application for Visualization of Air Pollution in a Big Cities

Bachelor’s thesis title in Czech:

Webová aplikace pro vizualizaci znečištění vzduchu ve velkých městech

Guidelines:

1) Gather requirement for the publicly available web application for visualization of air pollution in big cities. An application will use aggregated publicly available data.
2) Introduce possible technologies capable of fulfilling all the requirements. Compare technologies and propose solution of the problem. Propose GraphQL interface.
3) Design and implement the application. If possible, propose and perform unit testing.

Bibliography / sources:

[1] Alex Banks, Eve Porcello, Learning React, O'Reilly Media, 2017.
[2] Wieruch Robin, The Road to GraphQL, 2018 [accessible online]

Name and workplace of bachelor’s thesis supervisor:

Ing. Stanislav Vítek, Ph.D., Department of Radioelectronics, FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **12.02.2019** Deadline for bachelor thesis submission: **24.05.2019**

Assignment valid until: **20.09.2020**

Ing. Stanislav Vítek, Ph.D.	prof. Mgr. Petr Páta, Ph.D.	prof. Ing. Pavel Ripka, CSc.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.

Date of assignment receipt | Student’s signature

Declaration

I declare that I have completed the presented thesis independently and that I wrote down all the used sources in accordance with the methodological instruction on ethical principles in academic theses.

In Prague, 24. May 2019

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržovaní etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 24. května 2019

Abstract

This thesis focuses on the implementation of the client side of the web application for Air2Day platform. The main idea is to study the software development lifecycle, which allows to perform task division, model the solution, and technologies overview in order to implement the modelled solution.

Keywords:

Software, software development lifecycle, client-server architecture, web, JavaScript, Typescript, HTML, CSS

Abstrakt

Tato práce se zaměřila na implementaci klientské stránky webové aplikace pro platformu Air2Day. Hlavní myšlenkou je studium životního cyklu vývoje softwaru, který umožňuje provádět rozdělení úkolů a modelovat řešení a přehled technologií s cílem implementovat modelované řešení.

Kličková slova:

Software, životní cyklus vývoje softwaru, klient-server architektura, web, JavaScript, Typescript, HTML, CSS

Contents

Introduction

The aim of this thesis is to cover the first iteration of the implementation of the client-side of a web application for Air2Day platform, the ambition of which is to compensate the lack of fine monitoring of the air pollution and inform people about the current and forthcoming states of the air quality, empowering them to live healthier lives and create more sustainable business.

The long-term achievement of this ambitious mission was possible with the aid of real-time tracking system of fine particles at human respiratory levels and across streets and roads of densely inhabited areas with state-of-the-art moving and static sensors and an open-API platform for own or 3rd party data processing.

The first iteration of the web application for Air2Day platform has to cover the representation of aggregated publicly available data. To accomplish the mentioned goal, the software will be considered as a product and its potential lifecycle will be researched in the next chapter. Based on the knowledge learned from the research, the implementation has to be estimated. The research on the technologies and libraries possible to be used in order to implement the web application will be done, as well, and it is covered in "Air2Day implementation" chapter.

1. Product development lifecycle

One of the most depreciated parts of software development, by the technical mindsets, is the management. The management refers to all “non-coding” processes needed for the establishment of the final product. These processes have to be described, as well, in order to define the way of efficient target-setting for the implementation of the technical solution.

Software development lifecycles may differ and have some individual steps, which are specific to the company and product. In this chapter, we will take a look at the common and necessary stages that are applicable to any software. The following lifecycle stages will be considered:

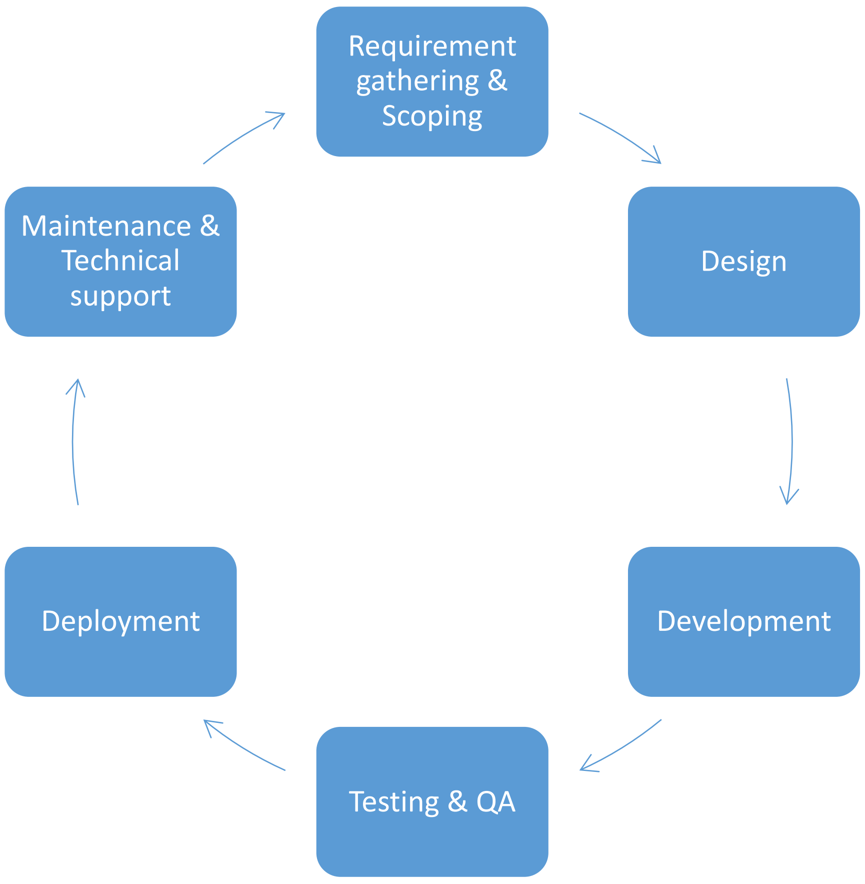

Requirement gathering & Scoping
Design
Development
Testing & QA
Deployment
Maintenance & Technical support

***Figure 1**. Software development lifecycle*

1.1. Requirements prioritization techniques and decision making

The first stage of the lifecycle is scoping and setting up the requirements, based on the client request.

"A requirement is a necessary attribute in a system, a statement that identifies a capability, characteristic, or quality factor of a system in order for it to have value and utility to a customer or user." (1)

In software development, the term *requirement* stands for a „must-have" functionality that should be performed. Setting well-defined requirements provides high flexibility and transparency of the development process by setting up exact tasks.

The aspects of prioritization that are usually considered are (2):

- *Importance*: multi-faceted term, but it stands, for example, for features which are must-have in the supposed product, or for those that require urgent implementation;
- *Penalty*: an estimate of the loss in case the requirement is not fulfilled;
- *Cost*: an estimate of the implementation cost supported by the developing company;
- *Time*: time frame needed for the implementation of the solution which fulfils the requirement;
- *Risk*: technical relevancy, market behaviour, external regulations, etc. are factors which also have an impact on the requirement;
- *Volatility*: very often taken as a part of the risk. It can also depend on business requirements changes, legislative changes or the requirement becoming clear only during the development process. It detrimentally affects planning and stability of the project.
- *Other aspects*: those are individual to the project. Competitors, release theme, financial benefit, etc.

The outcome of the requirement prioritization process is usually a scale; for the least robust techniques it is a nominal or ordinal scale and for the more powerful it is a ratio scale, due to the fact that it allows you to see not just the importance order, but also how one task is more important than the other. (2)

Numerous prioritization techniques exist, some of the most well-known in software development are presented in the subchapters bellow.

1.1.1. MoSCoW method

This method was developed by the Dai Clegg in 1994 (3). It has gained popularity because of its simplicity, universality, and suitability for the small projects. The outcome of this method is a nominal scale, where the requirements are grouped in four classes of different priority, the importance of the requirements inside of the class being equal. The name of the method is an acronym derived from the name of the classes.

The classes are the following (4):

1. *Must have*

   Each and every MUST requirement is considered critical and has to be implemented first. If at least one requirement from this class remains unfulfilled, it leads to the failure of the project.
2. *Should have*

   Set of the important, but not critical requirements, the unfulfillment of which remains unpleasant, but does not lead to failure.
3. *Could have*

   'Nice to have' features, which will be postponed first in case of timescale collapse during development.
4. *Won't*

   This class is minor for the prosperity of the project and definitely *won't* be fulfilled until the next deadline.

The main disadvantage of this method is the subjectivity in the assessment and the absence of any individual value assigned, which could lead to inaccurate results.

1.1.2. Cost-value approach

One of the most popular methods of requirement prioritization is the *cost-value approach*, created by Joachim Karlsson and Kevin Ryan. The main idea is to assign to each requirement its cost of implementation and the value that the requirement has. This method involves five steps (5):

b) Requirements review, to ensure they are comprehensive and clear;

c) Customers or users perform a cost/value estimation of the requirements based on AHP's[1] pairwise comparison;

d) Developers perform a cost/value estimation of the requirements based on AHP's pairwise comparison;

e) Developers create a diagram with *value* on the y-axis and *cost* on the x-axis;

f) Based on cost-value, the diagram project manager (stakeholder) makes a decision about implementation.

The steps described above include the so-called ***Analytic Hierarchy Process (AHP)*** also known as a mathematical tool for a systematic approach to complex decision-making problems.

AHP does not prescribe to the decision maker (DM) any "correct" decision but allows them to interactively find an option that is in the best way consistent with his understanding of the nature of the problem and the requirements for the solution.

This method was developed by the American mathematician Thomas L. Saaty, who wrote books about this method, developed software products, and has been conducting ISAHP [2] symposiums for 20 years. AHP is widely used in practice and actively developed by scientists around the world. Along with mathematics, it is also based on psychological aspects. AHP allows us to structure the complex problem of decision making in the form of a hierarchy in a clear and rational way, to compare and quantify alternative solutions. The method of analysing hierarchies is used throughout the world to make decisions in a variety of situations: from management at the interstate level to solving industrial and private problems in business, health care, and education.

[1] Analytic Hierarchy Process
[2] International Symposium on Analytic Hierarchy Process

AHP consists of four steps (6):

1. *Create $n \times n$ matrix of n requirements.*

   For example, assuming that we have four requirements to be estimated; by inserting those candidates into rows and columns we are getting a matrix of order 4×4.
2. *Evaluate relative intensity requirements with the aid of pairwise comparison on each criterion.*

   Evaluation based on scale represented in *Table 1*.

Relative intensity	Definition	Explanation
1	Of equal value	Two requirements are of equal value
3	Slightly more value	Experience slightly favors one requirement over another
5	Essential or strong value	Experience strongly favors one requirement over another
7	Very strong value	A requirement is strongly favored and its dominance is demonstrated in practice
9	Extreme value	The evidence favoring one over another is of the highest possible order of affirmation
2, 4, 6, 8	Intermediate values between two adjacent judgments	When compromise is needed
Reciprocals	If requirement *i* has one of the above numbers assigned to it when compared with requirement *j*, then *j* has the reciprocal value when compared with *i*.	

Table 1. *Scale for pairwise comparison. Taken from (7)*

Matrix filled in row to column comparison order. For example, in the beginning Req1 compared to Req1, relative intensity is equal, field filled with 1. Then Req1 compared to Req2, relative intensity of Req2 is "slightly more", field filled with $1/3$, and so on. The matrix could be the following:

	Req1	**Req2**	**Req3**	**Req4**
Req1	1	1/3	2	4
Req2	3	1	5	3
Req3	1/2	1/5	1	1/3
Req4	1/4	1/3	3	1

Table 2. *Requirement comparison matrix*

3. *Estimated eigenvalues of the comparison matrix.*

   This step consists of the normalization of each column, calculation of the sum of the rows, and the normalization of the sum (i.e. divide sum by the number of requirements). The results can look like this:

	Req1	**Req2**	**Req3**	**Req4**	**SUM**	**Normalized SUM**
Req1	0,21	0,18	0,18	0,48	**1,05**	**0,26**
Req2	0,63	0,54	0,45	0,36	**1,98**	**0,50**
Req3	0,11	0,11	0,09	0,04	**0,34**	**0,09**
Req4	0,05	0,18	0,27	0,12	**0,62**	**0,16**

Table 3. *Estimated eigenvalues of the comparison matrix*

4. *Set the relative values of the requirements according to the normalized sum.*

   Continuing the above example, the following total values of the requirements are obtained [1]:

Req1:	26%
Req2:	50%
Req3:	9%
Req4:	16%

[1] Note, the sum of the percentage result is 101%, while it should be 100%. This is given by the rounding off of the matrix's eigenvalues and is not an error.

It is worth mentioning that AHP is the most time-consuming part of this approach. Sometimes the cost-value approach is used in a simplified manner: the position of the requirement on the diagram depends on the subjective comparison of the requirements to each other and the agreement of the stakeholders. In this case, the method loses the benefit of the ratio scale and becomes more judgemental, but also less laborious.

1.2. Design

The design is a flexible stage and its definition depends on the nature of the developed software.

As it was mentioned in the introduction, the scope of this thesis is to implement the client-side of the web application. Before going into details, related specifically to the client-side design, I would like to recall basic concepts of software engineering.

The *Client-server* term refers to the software architecture model, which basically describes the existence of the provider of the service, called *server*, and the consumer of this service or the requester, called *client* (8). In modern web jargon, the server side is called “Back-end” and client side called “Front-end”.

The *Back-end* usually includes three major parts:

- the database, which stores the data;
- the server, which receives the requests;
- the application, which processes the requests, fetches data from the database, possibly processes the data and sends a response on request;

The request formats are unified by an API[1]. The most popular API architectures are SOAP[2] , REST[3] , and GraphQL, which is rather a specification, than an architecture and which has gained popularity in recent years.

In terms of back-end design, the stage could be briefly represented by the following steps:

- High-level design:

[1] Application Programming Interface
[2] Simple Object Access Protocol
[3] Representational State Transfer

    - in terms of database: identify tables and key elements;
    - in terms of server-side software: identify modules, draft their functionality, relations and dependencies between modules;
- Low-level design:
    - in terms of database: define the type and size of the tables;
    - in terms of server-side software: define the module logic and the functionality along with inputs and outputs;

Front-end is an interface between the user and back-end, which means that Front-end responsibilities usually include:

- interaction with the user;
- sending the request to back-end;
- representation of the received response;

The interaction with the user implies UI[1]. UI has its own set of specifications, which cover actions and interactions the end user can perform (9). The UI has to consider the usability; for this another design pattern, called UX[2], stands.

Front-end and back-end are equally important parts in terms of end-state, but in terms of software, they are separated products with different approaches. While back-end is strictly set on serving information efficiently, front-end has to consider not only the flow of the data received from back-end, but also the variety of subjective factors related to user perception.

Jean Kaiser assigned the following goals for the design stage of front-end (10):

- *Simplicity*

"Just because you can, doesn't mean you should." Jean Kaiser

Web pages from the late 90's and early 00's are some of the best examples of overwhelming the user with redundant animations, music in the background and a colour palette denser than a rainbow. The lack of designers in development

[1] User Interface
[2] User experience

departments leads to the same problems today. But since minimalism is in fashion, and a variety of ready-made templates exist, the situation is better.

- *Consistency*

  From the user point of view, the web application is a physical place, navigating from one page to another and should lead to a feeling of a closed system. Each and every page has to be part of the same template and act in the same manner, the colours should follow one scheme, and text content should have consistent fonts.
- *Identity*

  The web app for a construction company will not be the same as web app for a music band. Consistency, not only inside of the template, but with the domain it was made for, is important as well. Set up an identity and your user groups;
- *Robustness*

  Based on the *identity,* re-evaluate your requirements; specific user groups can bring additional expectations. Make sure your content is robust enough;
- *Navigability*

  Use best practices; it’s not a good time use your brilliant imagination and reinvent a wheel. The web app navigation has to be intuitive and positioned in a predictable location, external links have to be visually identified, cursors have to change accordingly;
- *Visual Appeal*

  If you achieved simplicity, add some aesthetic value. Pay attention to the look and feel of the content you provide, do it in the most natural way. Keep the interface simple and intuitive, with a focus on content – this strategy never fails;
- *Compatibility*

  Based on the *identity,* define the environments the application should be compatible with.

Based on the described goals, the front-end design requires numerous skills from a single developer. This is why, in commercial product development, the responsibilities are divided between the UI designer, the UX designer and the front-end developer.

Roger S. Pressman in his book “Software Engineering: A Practitioner's Approach” (10) offers to follow the design pyramid in Figure 2, where the levels represent design actions.

The first two levels represent the platform design, the selection of technology, and the application architecture. The other levels are about filling the skeleton with actions and content. The outcome of this process is usually wireframes or prototypes of future UI, with so-called user stories, which describe the behaviour of particular parts of an interface.

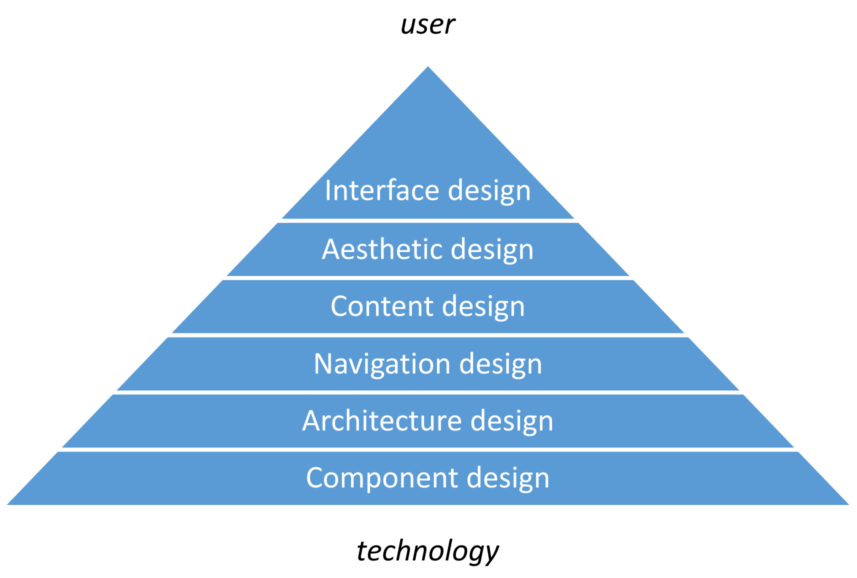


Figure 2. *Design Model. Take from (10)*

1.3. Development

In this stage, the actual implementation starts. If the previous stages performed properly and the requirements were transformed into user stories without loss, the coding should be painless. Unfortunately, such an ideal situation is very rare. Even the most experienced specialist can fail to predict some details, leading to the correction of the tasks on the fly, or in the worst situation, to the reimplementation of some functionality. The outcome of this stage is the build of the product that could be deployed.

The development process could be committed according to some specific model. The model is kind of a framework that the development team chooses individually. The most popular models these days are Waterfall and Agile.

1.4. Testing & QA[1]

Testing is usually a subcategory of the *Design* and *Development* stages. The wireframes or prototypes have to be tested in terms of intuitiveness and user-friendliness. The code created during development is usually written along with tests for particular modules. The UI can be tested manually or with automated tests.

The final testing had to check the whole build and assure the absence of critical issues, otherwise, the product is not ready for delivery to the customer.

1.5. Deployment

After the testing and possible error/bug fixing are done, the formal release and product deployment occur.

1.6. Maintenance & Technical support

The main goal is to ensure that the customer has no obstacles to use application. Once the product is deployed, it has to be monitored. While the customers are using the product, possible problems and bugs can occur and have to be fixed. If the product nature requires it, the current version will be upgraded or enhanced.

[1] Quality assurance

2. Air2Day implementation

During the summer semester of 2019, the first iteration of Air2Day was implemented. The high-level idea of this implementation was the following:

- aggregate data about air pollution from static sensors of known locations (done by the chief scientist of the project);
- implement back-end with the database based on the aggregated data and GraphQL API (had to be done by the Master's degree student involved in the project);
- implement front-end that communicates with the GraphQL API and represents the provided data (scope of this thesis);

2.1. Validation

The front-end implementation started with target-setting, i.e. requirement gathering, and defining of so-called user-stories (i.e. scenarios of how the requested requirement should be fulfilled through the application's performance). The fictional stakeholders are the *user* and the *developer.* The developer was used with the aim of describing maintenance and technical tasks, even though it is not a usual persona. Connextra template, recommended by Mike Cohn in his book "User Stories Applied: For Agile Software Development", was used in order to perform the feature description (11):

"I as a <role> want <function> so that <value>"

Even though some of the user stories may seem obvious, the clarification of the reasons, why a particular feature is on the to-do list instead of others, sometimes brings not only a deeper understanding of product objectives, but even new requirements.

The following results were produced:

a) Compatibility with GraphQL API
   Story: *"I, as a developer, want to have compatibility with the GraphQL API client, so that data could be fetched from the server."*
b) The representation of sensor locations on the map has to be included

Story: *"I, as a user, want to see sensors on the map, so that I know where the data collecting units are located."*

c) A table representation of the gathered data has to be included

Story: *"I, as a user, want to see a table representation of the gathered data, so that the content is clear and well-arranged."*

d) A chart representation and of the data has to be included

Story: *"I, as a user, want to see a chart representation of the gathered data, so that I have a picture of the pollution trends."*

e) Instruments for data filtration by time periods, locations, etc.

Story: *"I, as a user, want to data filtration be available, so that I'm able to setup graphs or tables according to the desired data."*

f) Maximize cross-browser compatibility

Story: *"I, as a user, want a maximum browser compatibility, so that I'm able to open web page in any browser available to me at the moment."*

g) Maximize uniqueness of the design

Story: *"I, as a developer, want my product design to be unique so that it will be remembered."*

h) Compatibility with mobile devices

Story: *"I, as a user, want the web application to be compatible with a mobile device, so that I'm able to open it when my laptop is not available."*

i) Accessibility

Story: *"I, as a user with disabilities, want the web application to be accessible so that I have the opportunity to be informed about air pollution."*

The order of task implementation had to be defined. Between the MoSCoW method and Cost-value approach, the first way of prioritization was used because of its simplicity, and results are represented in Table 4:

MUST	SHOULD	COULD	WON'T
a	d	e	g
b		f	i
c		h	

Table 4. *Prioritized requirement for Air2Day implementation*

As it can be seen in the last column, in the current release, I have no intention to focus on the design but assure basic functionality and rule out the accessibility goal. It is worth mentioning, that **accessibility is a MUST** long-term goal for the final implementation, and it is ruled out here only because this version will not be available to users.

2.2. Prototyping

Based on the drawn-up user-stories, first pencil sketches, and then prototypes were created using the online service Figma. For the prototyping of the desktop version, a MacBook Pro preset with 1440x900 resolution was used and for the mobile version, an iPhone SE preset with 320x568 resolution was used, with the assumption that it is one of the smallest devices on the market today. In order to achieve the best user experience, the following differences between the mobile and the desktop layout exist. The desktop layout was divided vertically and the map was placed on the left half of the display. The mobile layout was divided horizontally and the map placed in the upper part. It was done with the aim to prevent inconvenience in the display while scrolling.

Currently, the existing requirements imply the existence of only two screens, but a navigation bar was created with the assumption of functionality extension.

The created prototypes have an auxiliary character of mock-ups because they are not set on design. A ready-made kit of UI components will be used, which will be discussed in the following chapter. The exported prototypes are available in 5.1.1. Attachment 1: Air2Day prototypes.

2.3. Front-end development technologies overview

The front-end was based on three pillars, i.e. coding languages:

- HTML[1] - used to create a structured skeleton of web page layout;
- CSS[2] - used to style the pieces created with the HTML structure;
- JS[3] - used to add functionality and interactivity;

[1] Hyper-Text Markup Language
[2] Cascading Style Sheets
[3] JavaScript

In order to give the back-end technology a *client-side,* rendering is required. What this means is that the browser receives the bare-bones of the HTML and JS files that will be rendered, the benefit of this technology being the high interface interactivity and probably, the only disadvantage is the longer initial load time. The client-side rendering approach became very popular in the development of JS frameworks and libraries. In comparison to the *server-side* rendering, a ready-made HTML with already filled data is received by the browser. This is a static approach, and the full page has to be reloaded. It also leads to frequent requests to the server.

2.3.1. JavaScript frameworks and libraries

The software framework defines the whole application design, while the libraries are functions that can be in the code. None of these technologies is preferred over the other because both of them are, theoretically, flexible enough and serve our needs.

In order to select one of the frameworks/libraries, the most popular were compared using the AHP technique mentioned in Cost-value approach description. The most popular frameworks/libraries to be compared: Angular, React, Ember.js and Vue.js.

Each framework/library was compared according to each of the criteria and the following results were obtained:

a) *Usage statistic (based on the received stars on the GitHub service on 20.05.2019):*

- Angular – 48 241 (12)
- React – 129 389 (13)
- Ember.js – 20 984 (14)
- Vue.js – 138 937 (15)

There was no need to calculate the estimated eigenvalues here, relative values were calculated directly: Angular – **14,3 %**, React – **38,3 %**, Ember.js – **6,2 %**, Vue.js – **41,2 %**.

b) *Compatibility with TS[1]:*

TS is a superset of JS with a static type checking, used in order to prevent possible type errors. Any of the JS frameworks should be TS compatible with additional

[1] TypeScript

compilers and tricks around. But of course, the best solution for us is official support or solution that does not require much effort and is maintainable in the future. The official documentation provided the following results:

- Angular (besides AngularJS, which is legacy solution) is a completely TS based framework (16);
- For React and Vue.js official documentation (17) (18) confirmed the support of TS and the existence of tools available for it;
- Information about TS support in Ember.js official documentation is very short and uninformative, no official tools provided (19);

The results of the AHP calculation presented in Table 5 and Table 6, converted to relative values results of comparison, are the following: Angular – **52 %**, React – **20 %**, Ember.js – **8 %**, Vue.js – **20 %**.

	Angular	React	Ember.js	Vue.js
Angular	1	3	5	3
React	1/3	1	3	1
Ember.js	1/5	1/3	1	1/3
Vue.js	1/3	1	3	1

Table 5. *Comparison matrix of TS compatibility*

	Angular	React	Ember.js	Vue.js	SUM	Norm. SUM
Angular	0.54	0.56	0.42	0.56	2.08	0.52
React	0.18	0.19	0.25	0.19	0.80	0.20
Ember.js	0.11	0.06	0.08	0.06	0.32	0.08
Vue.js	0.18	0.19	0.25	0.19	0.80	0.20

Table 6. *Estimated eigenvalues of the comparison matrix of TS compatibility*

c) Availability of work experience with framework/library:

This is the most subjective criterion to compare, but still very important. The absence of the need to learn significantly increases the developer’s efficiency and leaves more time for feature development instead of discovering the main concepts of the technology.

The results of the AHP calculation presented in **Table 7** and **Table 8**, converted to relative values results are: Angular – **25 %**, React – **55 %**, Ember.js – **10 %**, Vue.js – **10 %**.

	Angular	React	Ember.js	Vue.js
Angular	1	1/3	3	3
React	3	1	5	5
Ember.js	1/3	1/5	1	1
Vue.js	1/3	1/5	1	1

Table 7. *Comparison matrix of available experience*

	Angular	React	Ember.js	Vue.js	SUM	Norm. SUM
Angular	0.21	0.19	0.30	0.30	1.01	0.25
React	0.64	0.58	0.50	0.50	2.22	0.55
Ember.js	0.07	0.12	0.10	0.10	0.39	0.10
Vue.js	0.07	0.12	0.10	0.10	0.39	0.10

Table 8. *Estimated eigenvalues of the comparison matrix of available experience*

d) Quality of documentation:

Even the best technologies could remain unknown if they are poorly described. Clear documentation is a prerequisite for a smooth learning curve. The estimation using this criterion is mostly subjective as well and based on personal impressions.

The documentation for Ember.js (19) was the only that left a bad impression because of the mentioned earlier lack of content. Clear release notes for each update and the recently added wide language support of the React documentation (20) deserve positive feedback. Vue.js provides descriptive documentation in 8 languages (18). No claims addressed to Angular documentation (16).

The results of the AHP calculation presented in **Table 9** and **Table 10**, converted to relative values results are: Angular – **16 %**, React – **50 %**, Ember.js – **8 %**, Vue.js – **26 %**.

	Angular	React	Ember.js	Vue.js
Angular	1	1/3	3	1/3
React	3	1	5	3
Ember.js	1/3	1/5	1	1/3
Vue.js	3	1/3	3	1

Table 9*. Comparison matrix of documentation quality*

	Angular	React	Ember.js	Vue.js	SUM	Norm. SUM
Angular	0.14	0.18	0.25	0.07	0.64	0.16
React	0.41	0.54	0.42	0.64	2.00	0.50
Ember.js	0.05	0.11	0.08	0.07	0.31	0.08
Vue.js	0.41	0.18	0.25	0.21	1.05	0.26

Table 10*. Estimated eigenvalues of the comparison matrix of documentation quality*

e) Maintainability

Long-term support for of the application can become very complicated if the update strategies of the framework-development team are not good or the application can simply get stuck in its development if no updates are provided over time.

For bad version compatibility, Angular is famous. The first two versions of the framework were not compatible. Over time, the cross version compatibility has improved but it still updates with significant changes and this is one of the reasons why Angular lost its audience.

Vue.js is the youngest framework that has only two generations. 90% of the API remains the same in the second version compared to the first according to documentation (21).

The react development team creates library updates very fast; upgrades go smoothly if you not unduly prolong it. Otherwise, an upgrade after a few major releases could be complicated. But it seems that that library continues to gain popularity and applications being developed using React are likely to be supported for a long time.

Ember.js future does not look as positive. The development team submits features slowly and the Ember audience migrates to other solutions.

The results of the AHP calculation presented in **Table 11** and **Table 12**, converted to relative values results are: Angular – **20 %**, React – **52 %**, Ember.js – **8 %**, Vue.js – **20 %**.

	Angular	React	Ember.js	Vue.js
Angular	1	1/3	3	1
React	3	1	5	3
Ember.js	1/3	1/5	1	1/3
Vue.js	1	1/3	3	1

Table 11. *Comparison matrix of documentation quality*

	Angular	React	Ember.js	Vue.js	SUM	Norm. SUM
Angular	0.19	0.18	0.25	0.19	0.80	0.20
React	0.56	0.54	0.42	0.56	2.08	0.52
Ember.js	0.06	0.11	0.08	0.06	0.32	0.08
Vue.js	0.19	0.18	0.25	0.19	0.80	0.20

Table 12*. Estimated eigenvalues of the comparison matrix of documentation quality*

Summary of the calculated results is next:

Criterion	Angular	React	Ember.js	Vue.js
a)	14.3 %	38.3 %	6.2 %	41.2 %

b)	52 %	20 %	8 %	20 %
c)	25 %	55 %	10 %	10 %
d)	16 %	50 %	8 %	26 %
e)	20 %	52 %	8 %	20 %
Average:	25.46 %	43.06 %	8.04 %	23.44 %

Table 13. *Summary of JS framework validation*

Based on the performed evaluation, I came to the conclusion that **React** is the best possible choice, not only because of the available personal experience, but also because of the high flexibility and the good prospect of this library.

2.3.1.1. React project setup

To get started with the project setup, first, one of the package management tools has to be installed on a local development machine. In our case ***Yarn*** was used. The installation guide can be found in the official documentation (22). Another compulsory tool is ***Node***, a JavaScript runtime environment version 8.10 and newer, available on the official web page (23).

React provides a quick starting solution to create a single-page application, called Create React App. It is available online as a package that could be installed with Yarn, npx or npm clients. This solution provides preinstalled and preconfigured tools such as Babel (24) and Webpack (25) inside, which means ready-made browser compatibility and compilation of the next generation JS into plain JS. A preconfigured version for TS of Create React App exist as well, this being used for the Air2Day project setup, with the following command:

```
yarn create react-app air2day –typescript
```

The created project can be run in the development mode using the following command:

```
yarn start
```

To add/remove another dependency (package), the following command has to be used:

```
yarn add package
yarn remove package
```

Where **package** is the name of the dependency you want to adjust.

For the navigation between the components of the application, the ***react-router*** package was added, along with its dependency ***react-router-dom***.

In the aim of the future localization support ***react-intl*** package was added. Any text that should be present on the page is defined as an object, where the **`id`** will be the key for a message in different languages. Example:

```
…
home: {
  id: "screen.appBar.home",
  description: "screen.appBar.home",
  defaultMessage: "Home"
},
…
```

2.3.2. GraphQL client side

In order to build a client compatible with GraphQL API, Apollo Client (26) was used. The following dependencies were installed:

- ***graphql***
- ***react-apollo***
- ***apollo-boost***
- ***apollo-client***
- ***apollo-link-context***

This client is easy to set up. It is enough to just wrap React app with Apollo Provider as it is demonstrated in the example below:

```
…
const httpLink = createHttpLink({
  uri: "your_uri",
  headers: {
    authorization: `your_autorization_header`
  }
});
const client = new ApolloClient({
  link: httpLink,
  cache: new InMemoryCache()
});

ReactDOM.render(
  <ApolloProvider client={client}>
    <App />
  </ApolloProvider>,
  document.getElementById("root")
);
…
```
`headers` is an optional property, needed only if your server requires authorization.

Another advantage of this client is the treatment of GraphQL queries as components. The “Query” component returns an object that contains loading, error, and data properties, and it allows us to manage the state and work on the UI in a very elegant way. Example:

```
…
<Query>
  {({ data, loading, error }) => {
    if (loading) {
      return <Loading />;
    }
    if (error) {
      return <ErrorBanner />;
    }
    if (data) {
        console.log(data);
    }
  }}
</Query>
…
```
GraphQL code generator (27) was used to setup the Apollo Client with TS addition library.

The following dependencies were added:

- ***graphql-codegen-typescript-client***
- ***graphql-codegen-typescript-common***

These packages require configuration, which in our case is the folllowing *codgen.yml*:

```
overwrite: true
schema: "./schema.graphql"
documents: "**/*.graphql"
config:
  noNamespaces: true
```

```yaml
generates:
  src/generated/graphql.tsx:
    plugins:
      - typescript
      - typescript-operations
      - typescript-react-apollo
```

In other words, the installed library will base the types on the `schema` property, i.e. file `schema.graphql`, the components will be generated from documents containing queries, i.e. with `*.graphql` name.

2.3.3. Data visualization library

Based on the same criteria as for JS framework selection in previous subchapter and the compatibility with React, the ***recharts*** (28) library was chosen for chart representations. The library has a rich API that allows building the following charts:

- Area chart
- Bar chart
- Line chart
- Composed chart
- Pie chart
- Radar chart
- Radial bar chart
- Scatter hart
- Treemap

Moreover, the components of this library are very flexible when it comes to styling. However, the default design is satisfactory as well.

2.3.4. Map provider

Between map providers, Google Map (29) has the most powerful API. Unfortunately, most of the libraries which aim to serve Google Maps API in React are very limited and do not cover what Google offers. In spite of that, in order to simplify the first implementation steps, ***google-map-react*** (30) was used to connect Google Map API to React.

The library to retrieve the addresses from latitude and longitude compatible with typescript does not exist yet. This forced me to make asynchronous requests directly to the Google Maps API. This complicates the state management and adds code-style inconsistency. Technically, this is not a problem but it is not an ideal situation either.

2.3.5. UI Kit

Material-UI (31) – a ready-made UI kit was used for the layout compassioning along with Google's Material Design guidelines. The advantages of this approach are:

- Satisfactory initial design;
- High flexibility of the design construction;
- Time saving and the possibility to focus on feature development;
- The UI kit was already tested, so upon agreement with the supervisor, we decided not to spend any time on rewriting tests;

3. Conclusion

During the work on this thesis, the first iteration of the client-side web application for Air2Day platform was implemented. With this purpose, the software development workflow was examined together with the decision-making strategies. Based on the knowledge gathered, the implementation was done.

According to the information provided by GitHub, the project is 93% of TypeScript, 5.0% HTML and 2% of CSS. The project was built using:

- React
- Apollo Client
- Material-UI
- Recharts
- Google Map React

The project guides (installation, deployment), along with the prerequisites are in the *read.me* file available in the root folder of the project.

The functionality implemented covers everything planned for the first iteration.

In the next iteration, Google Map React library will be replaced in favor of our own implementation. This decision was made because of the inconvenience caused by the library API and the very limited coverage of the Google Map API. There are only a few libraries for React that work with Google Map API and all of them are badly supported and very limited. Otherwise, the development went smoothly and the release of the next version is planned for the middle of July 2019.

4. References

1. Young, Ralph R. *The Requirements Engineering.* Norwood : ARTECH HOUSE, INC, 2004. 1-58053-266-7.

2. *The Fundamental Nature of Requirements Engineering Activities as a Decision Making Process.* Aybüke Aurum, Claes Wohlin. 14, Sydney, Australia; Ronneby, Sweden : Elsevier, 2003, Vol. 45. 000185957600002.

3. Clegg, Dai and Barker, Richard. *Case Method Fast-Track: A Rad Approach.* Boston : Addison-Wesley Longman Publishing Co., Inc, 1994. 020162432X.

4. *HYBRID APPROACH FOR REQUIREMENT PRIORITIZATION.* Sherraz, Ahmed, Abbas, Ahmad and Haad, Ali. 12, s.l. : IJSER, 2018, International Journal of Scientific & Engineering Research, Vol. 9. 2229-5518.

5. *A Cost–Value Approach for Prioritizing Requirements.* Karlssonn, Joachim and Ryan, Kevin. 5, s.l. : IEEE, 1997, Vol. 14. 1937-4194.

6. *Relative measurement and its generalization in decision making why pairwise comparisons are central in mathematics for the measurement of intangible factors the analytic hierarchy/network process.* Saaty, Thomas L. s.l. : Springer-Verlag, 2008. 1579-1505.

7. *The Analytic Hierarchy Process.* T.L.Saaty. New York : McGraw-Hill, 1980.

8. Oluwatosin, Haroon Shakirat. *Client-Server Model.* Sintok : IOSR Journal of Computer Engineering, 2014. Vol. 16.

9. Galitz, Wilbert O. *The Essential Guide to User Interface Design.* Toronto : Robert Ipsen, 2002. 0-471-084646.

10. Pressman, Roger S. *Software Engineering: A Practitioner's Approach.* s.l. : Palgrave Macmillan, 2005. 978-0-07-301933-8.

11. Cohn, Mike. *User Stories Applied: For Agile Software Development.* s.l. : Addison-Wesley Professional, 2004. 978-0321205681.

12. angular. *GitHub.* [Online] [Cited: may 20, 2019.] https://github.com/angular/angular.

13. react. *GitHub.* [Online] [Cited: may 20, 2019.] https://github.com/facebook/react/.

14. ember.js. *GitHub.* [Online] [Cited: may 20, 2019.] https://github.com/emberjs/ember.js.

15. vue. *GitHub.* [Online] [Cited: may 20, 2019.] https://github.com/vuejs/vue.

16. Docs. *Angular.* [Online] [Cited: may 20, 2019.] https://angular.io/docs.

17. Adding TypeScript. *Create React App.* [Online] [Cited: may 20, 2019.] https://facebook.github.io/create-react-app/docs/adding-typescript.

18. TypeScript Support. *Vue.js.* [Online] [Cited: may 20, 2019.] https://vuejs.org/v2/guide/typescript.html.

19. Web Development. *Ember.* [Online] [Cited: may 20, 2019.] https://guides.emberjs.com/release/glossary/web-development/#toc_coffeescript-typescript.

20. Languages. *ReactJS.* [Online] [Cited: may 20, 2019.] https://reactjs.org/languages.

21. Migration from Vue 1.x. *Vue.js.* [Online] [Cited: may 20, 2019.] https://vuejs.org/v2/guide/migration.html.

22. Installation. *Yarn.* [Online] [Cited: may 22, 2019.] https://yarnpkg.com/en/docs/install#mac-stable.

23. Home. *Node.js.* [Online] [Cited: may 22, 2019.] https://nodejs.org/en/.

24. Home. *Babel.* [Online] [Cited: may 22, 2019.] https://babeljs.io/.

25. Home. *Webpack.* [Online] [Cited: may 22, 2019.] https://webpack.js.org/.

26. Introduction. *Apollo Docs.* [Online] [Cited: may 20, 2019.] https://www.apollographql.com/docs/react/.

27. GraphQL Code Generator. *GitHub.* [Online] [Cited: may 20, 2019.] https://github.com/dotansimha/graphql-code-generator.

28. Home. *Recharts.* [Online] 2016. [Cited: may 20, 2019.] http://recharts.org/en-US/.

29. Maps JavaScript API . *Google Maps Platform.* [Online] [Cited: may 20, 2019.] https://developers.google.com/maps/documentation/javascript/tutorial.

30. Google Map React. *GitHub.* [Online] [Cited: may 20, 2019.] https://github.com/google-map-react/google-map-react.

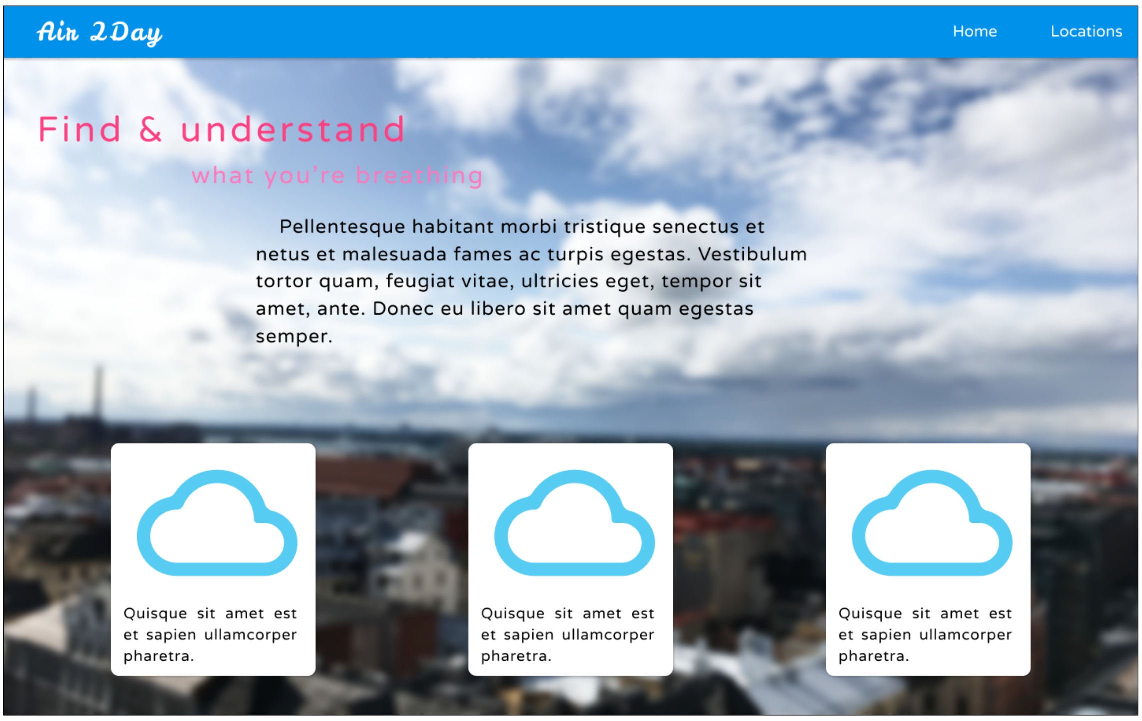
31. MATERIAL-UI. *Home.* [Online] [Cited: may 20, 2019.] https://material-ui.com.

5. Attachments

5.1. List of attachments

- Attachment 1: Air2Day prototypes
- Attachment 2: File air2day.zip

5.1.1. Attachment 1: Air2Day prototypes



Find & understand

what you're breathing

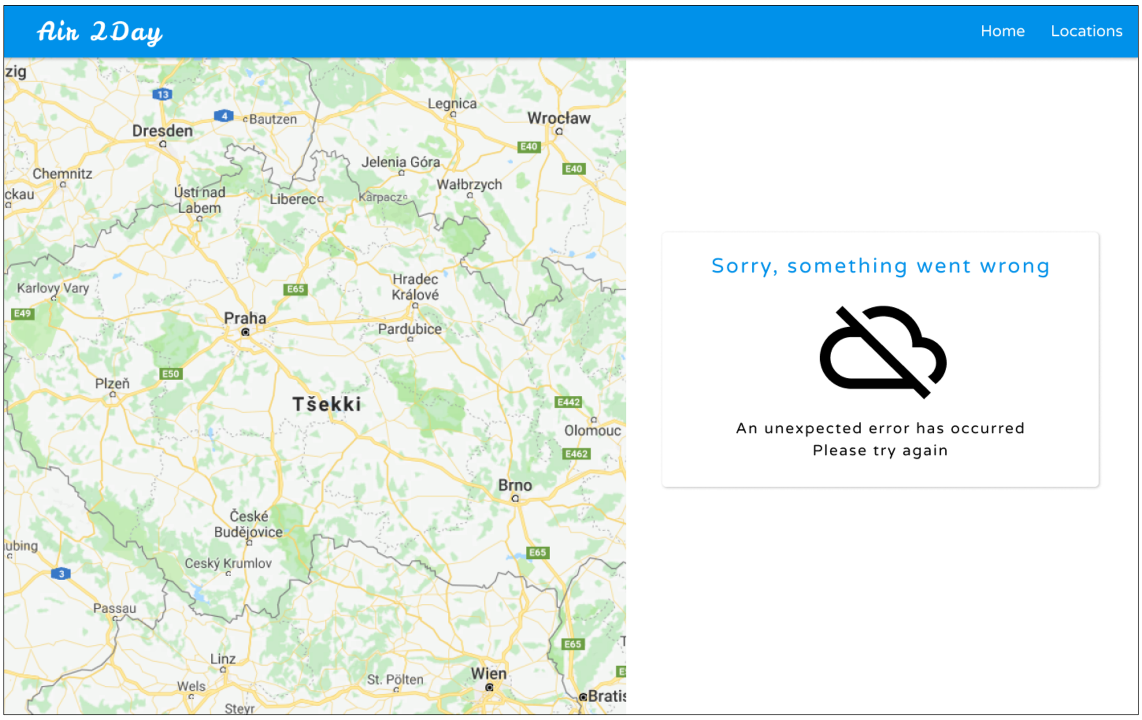
Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper.

Quisque sit amet est et sapien ullamcorper pharetra.

Quisque sit amet est et sapien ullamcorper pharetra.

Quisque sit amet est et sapien ullamcorper pharetra.

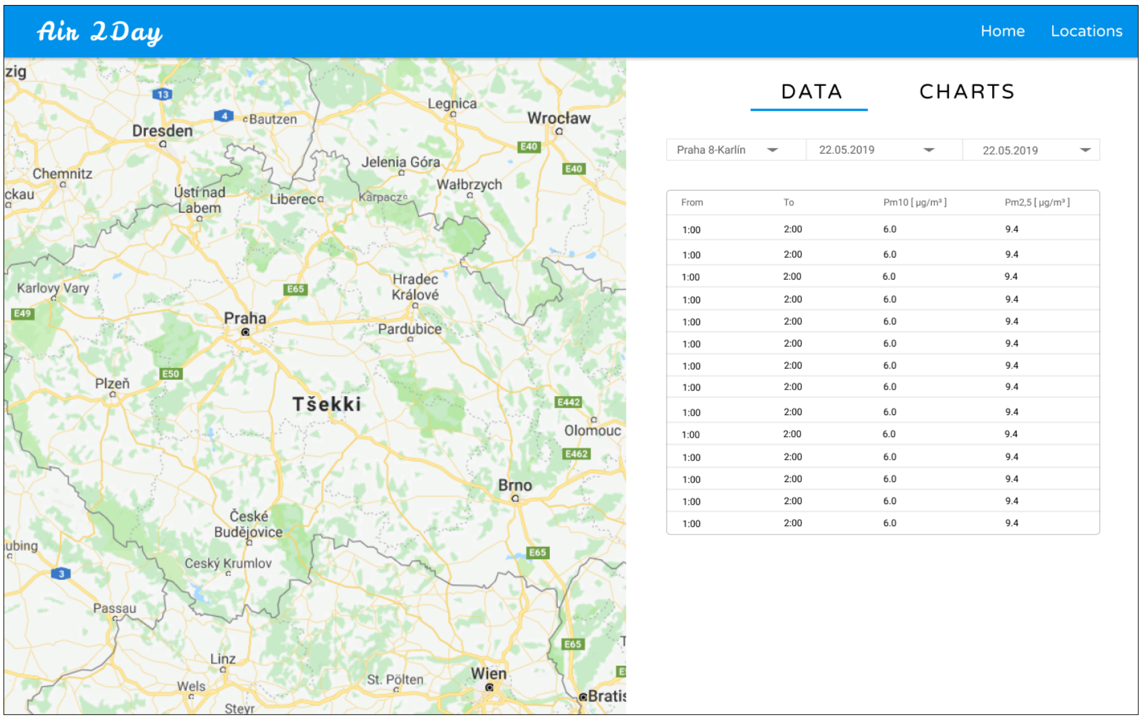
Figure 3. *Mock-up of landing page. Desktop size*



Sorry, something went wrong

An unexpected error has occurred
Please try again

Figure 4. *Mock-up of error banner. Desktop size*



From	To	Pm10 [μg/m³]	Pm2,5 [μg/m³]
1:00	2:00	6.0	9.4
1:00	2:00	6.0	9.4
1:00	2:00	6.0	9.4
1:00	2:00	6.0	9.4
1:00	2:00	6.0	9.4
1:00	2:00	6.0	9.4
1:00	2:00	6.0	9.4
1:00	2:00	6.0	9.4
1:00	2:00	6.0	9.4
1:00	2:00	6.0	9.4
1:00	2:00	6.0	9.4
1:00	2:00	6.0	9.4
1:00	2:00	6.0	9.4
1:00	2:00	6.0	9.4

Figure 5. *Mock-up of data representation. Desktop size*

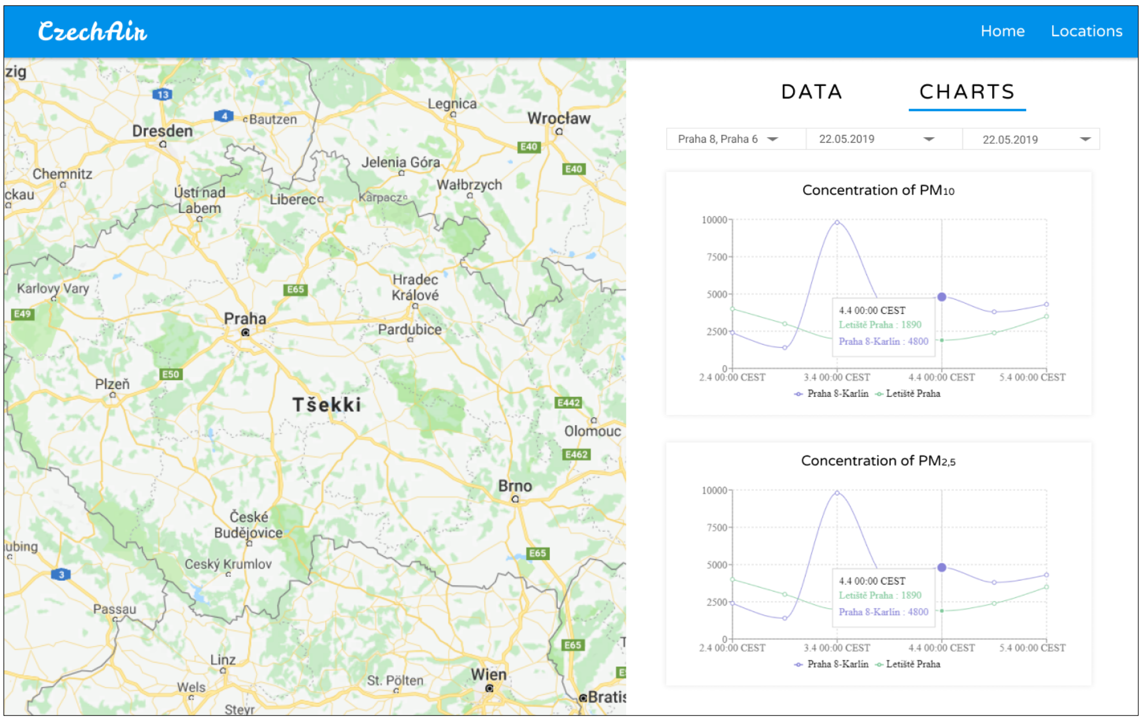
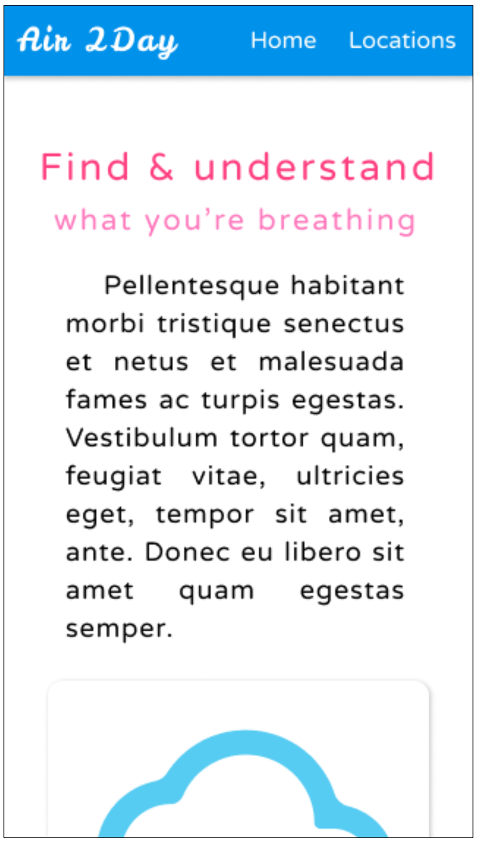


Figure 6. *Mock-up of chart representation. Desktop size*

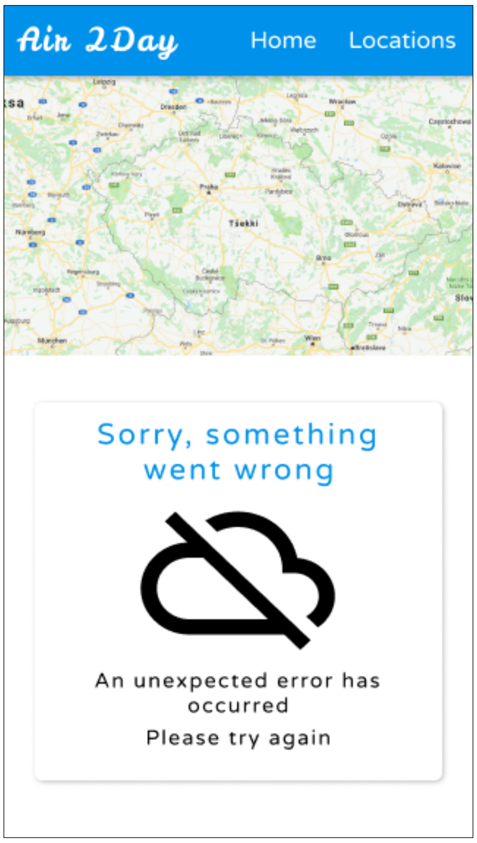

Air 2Day Home Locations

Find & understand

what you're breathing

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper.

Figure 7*. Mock-up of landing page. Mobile version*

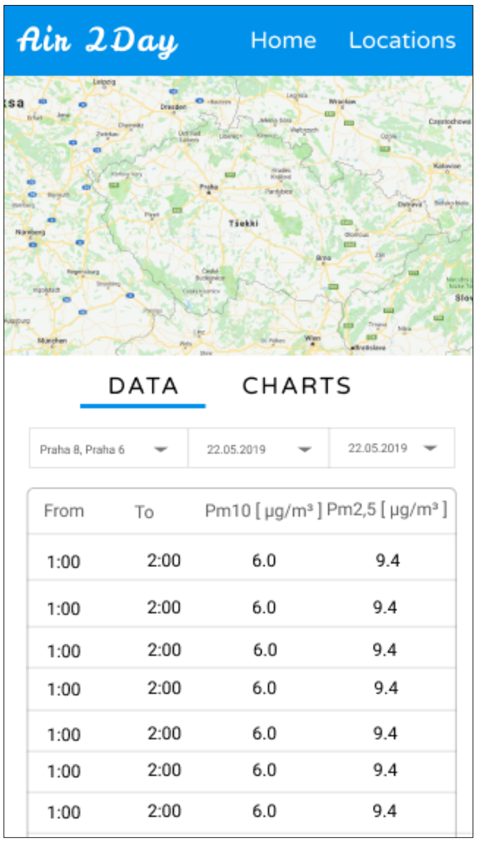

Air 2Day Home Locations

Sorry, something went wrong

An unexpected error has occurred

Please try again

Figure 8. *Mock-up of error banner. Mobile version*



From	To	Pm10 [μg/m³]	Pm2,5 [μg/m³]
1:00	2:00	6.0	9.4
1:00	2:00	6.0	9.4
1:00	2:00	6.0	9.4
1:00	2:00	6.0	9.4
1:00	2:00	6.0	9.4
1:00	2:00	6.0	9.4
1:00	2:00	6.0	9.4

Figure 9. *Mock-up of data representation. Mobile version*

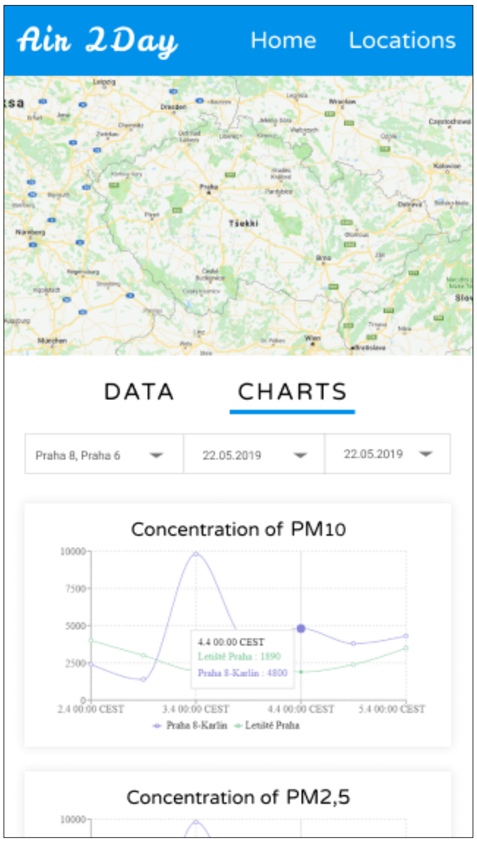


Figure 10. *Mock-up of chart representation. Mobile version*

5.1.2. Attachment 2: File air2day.zip

Contains React project files along with a read.me file with installation instructions.