Theory of Hybrid Systems Informatik 2 | RWTH AACHEN UNIVERSITY

**Diese Arbeit wurde vorgelegt am**
**Lehr- und Forschungsgebiet Theorie der hybriden Systeme**

# Visualization of the Noise Propagation in Wind Farms

## Visualisierung der Schallausbreitung von Windparks

Bachelorarbeit
Informatik

März 2021


| | |
|---|---|
| Vorgelegt von<br>Presented by | Aleksandar Timanov<br>Pontstraße 176<br>52062 Aachen<br>Matrikelnummer: 357632<br>aleksandar.timanov@rwth-aachen.de |
| Erstprüfer<br>First examiner | Prof. Dr. rer. nat. Erika Ábrahám<br>Lehr- und Forschungsgebiet: Theorie der hybriden Systeme<br>RWTH Aachen University |
| Zweitprüfer<br>Second examiner | Prof. Dr. rer. nat. Thomas Noll<br>Lehr- und Forschungsgebiet: Software Modellierung und Verifikation<br>RWTH Aachen University |
| Betreuer<br>Supervisor | Dr. rer. nat. Pascal Richter<br>Lehr- und Forschungsgebiet: Theorie der hybriden Systeme<br>RWTH Aachen University |

# Eigenständigkeitserklärung

Hiermit versichere ich, dass ich diese Bachelorarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Stellen meiner Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen sind, habe ich in jedem Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht. Dasselbe gilt sinngemäß für Tabellen und Abbildungen. Diese Arbeit hat in dieser oder einer ähnlichen Form noch nicht im Rahmen einer anderen Prüfung vorgelegen.

Aachen, im März 2021

Aleksandar Timanov

# Contents

# 1 Introduction

Over the last century, our environment has been changed drastically. We have observed a rise in temperatures which brings some severe consequences with it. This change in the climate affects all regions around the world and poses a threat to the economy, human health, entire ecosystems and other factors that influence our lives. More concretely lately many extreme weather events were observed, like for example the excessive ice melting on the polar ice sheets which result in a rise of sea levels [40, 2] and an increasing number of floods in cities around the globe [57, 41], or the drought in some other regions where an unusual number of wildfires have happened in the last years [48].

Humans have a large influence on climate change due to the excessive production of greenhouse gases such as carbon dioxide. The main source of these gases is the burning of fossil fuels such as coal, oil, and natural gas. This increase in the concentration of greenhouse gases in the atmosphere leads to heat being trapped when radiating from Earth towards space thus producing the so-called "greenhouse effect" [58].

Major sources of greenhouse gas emissions are the electricity production (27.5 percent of 2017 greenhouse gas emissions [70]) due to the very high part of the electricity being generated by burning fossil fuels like coal or natural gas (over 60 percent of the worlds energy production in 2015) [6]. The recent development of the part of electricity produced by fossil fuels is shown in Figure 1.

Lowering electricity production from fossil fuels urges into an increase in the amount of renewable energy sources [65]. Among the best natural resources of energy is the wind and because of that more and more wind farms are being built around the world in order to convert this energy into electricity, see Figure 2.

Although wind farms are having a good effect on climate, they also introduce some other problems mostly towards the people who have to live near a wind farm. Recently an increasing local resistance can be observed against the expansion of wind power in Germany. The main reasons for this are its impacts on the landscape [55], the removal of forests in order to open up space for the building sights, the emissions of low-frequency noise from the turbines [62, 46, 50] as well as a negative impact on the local wildlife [37, 72]. Oftentimes people, living nearby from a wind farm, report about the undesired and bothersome noises, shades or overall visual impairment on the landscape as a result from the wind power devices.

Most people are afraid of the mentioned drawbacks of wind farms and therefore protest against their construction [64, 68]. In most cases, however, they would not, in reality, be affected by these negative effects. Building a wind farm costs a lot of resources and many stakeholders depend on the success of the projects. Because of these reasons, it is highly inefficient for projects to be delayed or even canceled due to public resistance. Our goal is therefore to present the people with a tool, which would inform them as accurately and realistically as possible so that they can assess the situation better and as a result be able to make a more justified decision about the construction of a wind farm.

We hypothesize that if participants could receive accurate information and could experience a simulation, representing the real environment around the planned wind farms,

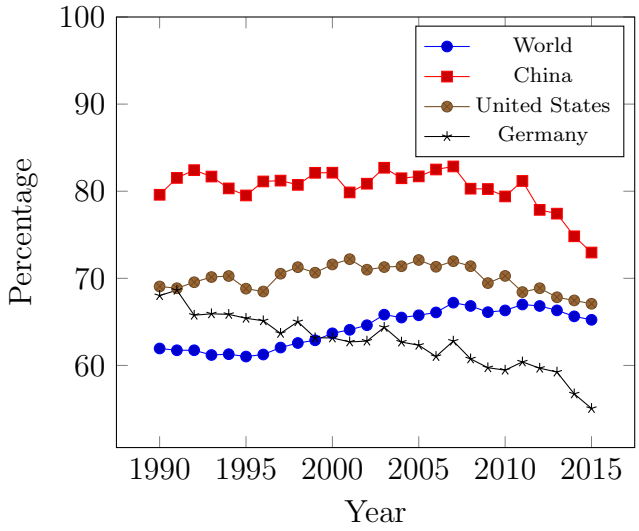



Figure 1: Electricity production from oil, gas and coal sources (% of total) [6].

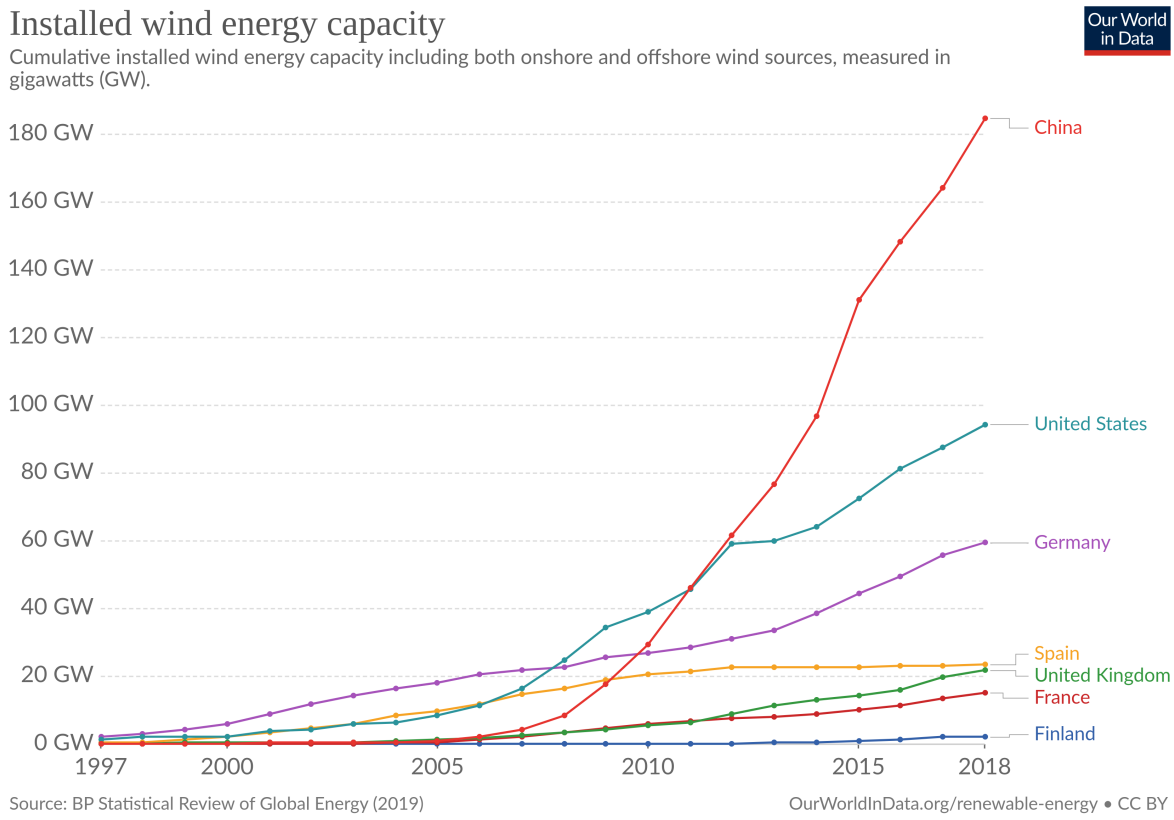



Figure 2: Cumulative installed wind energy capacity including both onshore and offshore wind sources, measured in gigawatts (GW)[65].

they would be able to take a more objective position about more of the planned wind farm projects.

We would like to introduce a way for the local population to be able to receive interactively presented information about the wind farms and in this way higher the acceptance among the people. Allowing them to participate in the projects with their own opinion and suggestions would also broaden their interest and engagement in the problems [54] so this is also an aim for us. In this work, we are presenting a software tool, which would ease the spread of information about the planned building sights for wind farms among the population. Our tool would attempt to allow it's users to experience a simulation of a planned building sight both in visual, as well as in aural aspects. This software is going to be available for mobile devices, as well as for a desktop computers, which would make it more approachable to the wider public.

To let the participants explore different scenarios it is planned for the application to allow the user to view the artificial objects from different perspectives and locations. They are going to be able to swap the types of turbines, weather conditions, times of the day and year and other aspects of the virtual surroundings as to observe different conditions and to get a firmer grasp of the way that the different turbines would work.

The application would also be directed towards creating a two-way information flow between the major investors in the projects and the public. The users would be able and encouraged to share their feedback regarding the projects or even participate in them as investors. Gathering such data would allow for better optimization of the planning process for future wind farms. For example, if the company behind a certain wind farm project observes that one type of turbine is more preferred by the local community in a certain region, then they might consider this type of turbine more often for future projects as it could limit the protests against the planed farms and thus lower the costs and speed up the construction process.

The returned feedback should be analyzed automatically afterward and be presented to the companies to inform them of the public opinion in the according regions. This transfer of data has to happen in a secure way which does not put the sensitive information of the participants at risk.

In order to synthesize and visualize some of the more technical aspects of a wind farms influence onto the surrounding areas, we had to develop our own algorithms. These algorithms would be used to take the raw quantitative data about the respective factor we would like to show and compute a suitable output, which would then be used for the graphical interface of our application.

One such algorithm, which is also the focus of this work, was the noise propagation approximation algorithm, that we have created. This algorithm takes into account factors like, meteorological conditions, positioning of the planned and already built turbines, geo-locations for objects which might influence the sound and others. It outputs a list of border line points configurations, with which we later on use to present the noise propagation from the wind farm into the surrounding area, as marked polygons on a map. These polygons show where the noise levels would potentially exceed noise level thresholds, set from us in advance.

## 1.1 Outline

Now that we have seen where the need for our work lies and how we can influence the situation in order to higher the acceptance of wind farms in this first section, we are going to further define the most important background information in the second section. There we are also going to explore some relevant research projects in the field.

After that in Section 3 we are going to present the methods we have used in building the noise propagation algorithm for our tool.

In Section 4 we show the outlines for the development of the frontend part of the tool itself, including the architecture idea, layout, the ideas for the different components themselves and how they would work and list all the technologies planned to be used for the implementation.

In Section 4 we are showing the results from our work and what the final product looks like and how it works.

In Section 5 we conclude how successful this project was and finally we propose some further development that might build upon the results of the current work.

# 2 State of the art

A wind farm is a group of wind turbines in the same location used to produce electricity. Wind farms vary in size from a small number of turbines to several hundred wind turbines covering an extensive area. They can also be divided into onshore and offshore wind farms, but we are only going to focus on the onshore ones, as they pose a bigger potential inconveniences for the local population.

Wind turbines are devices, that convert wind energy to electrical energy. Conventional horizontal axis turbines can be divided into the rotor, the generator, and the surrounding structure.

Choosing the type of turbine is determined by the wind class, the chosen location for the building sight falls into. The IEC Wind Classes show which turbines would be most productive and within the safety bounds and is calculated out of the average annual wind speed in this location, the extreme gusts that could occur within a 50 year period and how turbulent the wind at the sight is like shown in Figure 3 [25, 47, 74]. After determining these factors the location is put in one of four classes, with class I being the highest wind power. Depending on the wind class, different design and size are appropriate for the wind turbine, that would be placed. For example, turbines in locations with lower wind power and bigger rotors would produce the same amount of electricity as one with smaller rotors but in a windier location.

Depending on the potential built area and the types of turbines, a wind farm might produce different amounts of electricity, so as a part of our tool we have also implemented an optimization algorithm that takes into account the surface area and the users preferences and outputs possibilities for wind farm configurations. This algorithm is not the focus of this work, but the outputs from it were used for the construction of the noise propagation estimation.

| IEC Wind Classes | | | | |
|---|---|---|---|---|
| | I (High Wind) | II (Medium Wind) | III (Low Wind) | IV (Very Low Wind) |
| Reference Wind Speed | 50 m/s | 42.5 m/s | 37.5 m/s | 30 m/s |
| **Annual Average Wind Speed (Max)** | **10 m/s** | **8.5 m/s** | **7.5 m/s** | **6 m/s** |
| 50-year Return Gust | 70 m/s | 59.5 m/s | 52.5 m/s | 42 m/s |
| 1-year Return Gust | 52.5 m/s | 44.6 m/s | 39.4 m/s | 31.5 m/s |

Figure 3: IEC Wind Classes show which type of blades and rotors are needed for the wind turbines planned for a certain location [25].

## 2.1 Auralization

For our needs we have to develop a tool, that is user friendly in it's design, so that people of all educational background can easily get informed and comprehend the consequences of a certain wind farm construction. At the same time we want to make an accurate simulation, which would appear as a believable representation of the reality and the users would trust that what they see and hear could also happen in the real world. Because we are aiming for such a wide target of end users, we would also like to make the tool available on an array of devices, so the graphical interface would have to be well suited both for bigger displays and for mobile devices.

There are several already conducted experiments which we are going to take a look at in this section in order to draw inspiration and avoid possible mistakes, discovered in these works.

Noise propagation is a field with wide array of choices when it comes to choosing an approach to computing an aural simulation. Recently, there is also a lot of research in the field of noise propagation in the specific case, when the noise is caused as a consequence from wind farm rotor movements.

An experiment very close to our needs was conducted, where the goal was, as with our work, to measure if there would be an increase in community engagement in the topic of wind farm construction [36], if the general public was introduced a visually and aurally realistic simulation of the planned wind farms. Their approach uses the Virtual Community Noise Simulator (VCNS), previously used for the simulation of aircraft noise propagation. Acknowledging the similarities in the builds of wind turbine rotors and aircraft wings, the researchers have developed an aural and visual simulation for a Virtual Reality device. This simulation was tested and evaluated empirically. The results show that users, who have experienced the virtual simulation have had better understanding and more objective opinion towards the planned wind farms. Their approach has however

some big differences, which do not comply with our needs. Firstly their tool can be presented only via a special equipment for the experiencing of virtual reality scenes, with special audio equipment as well. This is not the case for us, we are aiming to develop a tool accessible on mobile devices and desktop computers. Because of this variety of different devices, it would be difficult to present similar experience and to ensure that the simulated environment would be realistic. Most devices have different audio output capabilities and thus would not produce the same aural effects, if we used a similar approach as the one, described above.

Normally noise propagation models are suitable for simple compositions of objects, emitters and receivers, and thus are not very efficient when applied to a large scale simulation like the ones we are trying to achieve, which take account of a whole region with all objects present there. As shown later on in the paper our approach was intended at optimizing the inputted data in order to transform our calculations into viable choice for simulating wind farm noise propagation. There is, however, different research projects on this topic, which explore other solutions to this problem. One such projects was developed to test the effect of raytracing methods in calculating sound propagation for large environments like the ones needed for a wind farm [39]. This approach reaches competitive runtimes on a personal computer, however their method does not take into effect the full effects of the atmospheric conditions and terrain which could affect the precision of the results.

## 2.2 Visualization

There are several already popular approaches and tools for visualizing wind farms. A research in the field of wind farm simulation was conducted by the UFZ-Helmholtz Center for Environmental Research [73]. They used a large scale projection-based visualization and virtual reality system which allows the user to perform simple visually-driven planning tasks like selecting and moving turbines, changing their type (from a predefined set of different turbine types) and removing them from the simulated landscape. Throughout the process, some constraints are also implemented, which represent real-world rules for building wind farms. For example, participants could not place turbines nearer to each other than their minimum spacing required for correct functionality. They could also choose the wind direction and speed. Afterwards feedback about the system was collected from the participants.

The main drawback of the tested system was that it was non-transportable, which meant that the participants had to specifically travel to the UFZ (in Leipzig) in order to take part in the discussion and experience the simulation. Another drawback, mentioned in their report, was the cost of the technologies used, which in most cases made the simulation hard for companies to afford. Both these negative sides were a product of the technological limits of the time. Since the conduction of the experiment in 2009 there were introduced great advances in the field of virtual visualization like for example the WebVR API (introduced in 2016) [33], which allowed for 3D interactive animations to be viewed on modern browsers. Even some modern game engines are being used for architectural planning for future buildings [59, 63, 56]. Hardware has also come a long way since then, eliminating the need for powerful stationary computers to do the heavy

calculations [42].

Another project, which is close to ours, developed an Augmented reality-based tool for visualizing wind farms [43]. This tool was developed with educational purposes in mind and is designed to help students in the field of renewable energy engineering to get a better grasp of their work. The application was running on mobile devices and showed planned wind turbines as a 3D overlay to the landscape currently seen through the device's camera. It used GPS navigation to determine the location of the user in comparison to the foreseen building sight thus presenting the turbines in their accurate size and location.

There are some similarities with this project and ours, for example in the way the tools are going to be distributed. Mobile devices are the main target of both applications.

However, there are some differences between the two projects as well. Our tool focuses not only on presenting the visual aspects of the wind farms, but also on their impact on the environment in terms of sound. Another difference is that their project was designed with only informational purposes in mind, we, on the other hand, would like to establish a discussion with the participants and allow them to present their own solutions or criticisms. We want to let the user experiment with different layouts and setups for the planned projects so that they could better understand how certain aspects could influence the environment.

Useful research was conducted using web-based simulations to determine which type of simulations of wind farms are better accepted by the public [38]. The types of visualization included a Zone of Theoretical Visibility (ZTV) Map, a Wireframe Diagram, realistic Photomontage and interactive, animated and still geographic information system (GIS)-Based 3D Landscape Visualization (LV). The experiment concluded that participants preferred the more traditional methods of visualization like Photomontage and the map, however, their main concern with the landscape visualization models was the usability, which means that bettering the responsibility and preventing the users from getting confused by the interface could bridge this gap.

The paper concludes that simulations are a possible way for information to be transferred to the wider public and suggest the building of a tool that introduces the models as a two-way flow of information between the participants and the companies behind the building projects.

There are also some tools that use simulations to visualize wind farms, but they are mainly aimed towards optimizing the productivity of the turbines and monitoring changes [28, 31]. However, in most cases, they are too complicated and not very user friendly, because they are targeting professionals with background in engineering, allowing them to observe the consequences of some theoretical change and compare the results with the currently available options. This makes them overall not well suited for our needs.

# 3 Noise Propagation Model

One of the aspects of a new wind farm project, which is a cause for concern in the residents, is the aural pollution which would be generated around the new turbines. The constant sound, produced by the rotation of the rotors, can cause discomfort for the people living near a wind farm. But how significant would be the sound when it reaches these residential areas? To answer this question we have developed an algorithm which approximates the levels of noise pollution, which would be generated for the regions around the planned wind farm. We are using one of the well-known noise propagation models, namely DIN ISO 9613-2 [45] to estimate the noise levels. Our results were tested to ensure their quality based on the test cases defined in DIN ISO/TR 17534-3 [45].

## 3.1 Noise propagation model (DIN ISO 9613-2)

Firstly we are going to take a closer look at how the noise propagation model DIN ISO 9613-2 works. The model itself is developed for the purposes of estimating any aural effects and is not specific to wind farms. This model takes as input the sound pressure levels of a source point, in our case a wind turbine, and various data related to factors, which affect spreading and dampening of the sound waves, and outputs the A-weighted equivalent continuous sound pressure level, estimated for different receiver points.

### 3.1.1 Basic equations

The first thing that has to be calculated is the sound power level $L_{\mathrm{fT}}(DW)$ for the receiver point for each octave frequency level with band centers between 63 Hz and 8 kHz. We do that by the following Equation Formula 1:

$$L_{\mathrm{fT}}(DW) = L_{\mathrm{W}} + D_{\mathrm{C}} - A \tag{1}$$

The octave band sound power level $L_{\mathrm{W}}$ of every point, in decibels (dB) can be derived directly from the sound frequency. Due to this relation we can use the already defined octave band levels from Table 1. This sound power level has to be adjusted however under the influence of the directive correction $D_{\mathrm{C}}$ and the dampening of the sound due to natural influences. The dampening can further on be described as the sum of the distortion levels, caused by each of 5 sources (Equation 2).

$$A = A_{\mathrm{div}} + A_{\mathrm{atm}} + A_{\mathrm{gr}} + A_{\mathrm{bar}} + A_{\mathrm{misc}} \tag{2}$$

More detailed information on the different approaches to calculating the dampening values is explored in the next paragraph (3.1.2 Dampening factors in detail). $A_{\mathrm{div}}$ refers to the dampening caused by geometric propagation. $A_{\mathrm{atm}}$ is the dampening caused by the absorption of the sound waves from the air. $A_{\mathrm{gr}}$ is caused by the ground effect of aerodynamics. $A_{\mathrm{bar}}$ is the dampening, which takes place when there is a barrier between the source point and the receiver point, which might distort the sound waves. In our case such barriers could be the buildings in the surrounding area around the wind turbine for example. Finally $A_{\mathrm{misc}}$ is left for all other effects which might affect the sound wave

| Frequency [Hz] | **63** | **125** | **250** | **500** | **1000** | **2000** | **4000** | **8000** |
|---|---|---|---|---|---|---|---|---|
| $L_{\mathrm{WA,P}}$ [dB] | 86,2 | 91,9 | 96,6 | 98,9 | 100,1 | 97,7 | 90,4 | 75,2 |
| $L_{\mathrm{WA,P}}$ [dB] | 84,3 | 89,9 | 92,6 | 94,3 | 94,0 | 91,7 | 83,6 | 64,0 |

Table 1: Octave band frequencies of wind turbines GE-5.3-158 NRO 105 (first row) [35] and Enercon E-138 EP3 / 3.500 kW with TES (second row) [34].

distribution in a negative way.

To calculate the average value for the A-weighted continuous sound pressure level in case of headwind, we can use Formula 3. This formula uses the sound power levels for all emitter points, calculated by Formula 1 with emitter point $i$ and index $j$ of the octave band frequency, and the dampening factors, calculated by Formula 2, in relation to the index $j$ of the octave band frequency and normalized according to the IEC 651 [44].

$$L_{\mathrm{AT}}(DW) = 10 \log \left( \sum_{i=0}^{N} \sum_{j=0}^{N} 10^{0{,}1(L_{\mathrm{FT}}(i,j)+A_{\mathrm{f}}(j))} \right) \tag{3}$$

This averaged out value for the continuous sound power level, can further on be adapted for calculations about the long term behavior, by taking account of the meteorological correction factor $C_{\mathrm{met}}$, like shown in Equation 4. This is the final value that we are going to use later on in our algorithm to evaluate borderline positions around the wind turbines.

$$L_{\mathrm{AT}}(LT) = L_{\mathrm{AT}}(DW) - C_{\mathrm{met}} \tag{4}$$

### 3.1.2 Dampening factors in detail

The geometrical dampening factor $A_{\mathrm{div}}$ takes into account the spherical propagation of the sound waves emitted by a source point into a free field. It is calculated by Equation 5, where $d$ is the distance traveled by the wave in meters and $d_0$ is the starting distance from the emitter.

$$A_{\mathrm{div}} = 20 \log \left( \frac{d}{d_0} \right) + 11 \tag{5}$$

The attenuation due to air absorption $A_{atm}$, calculated by Formula 6, is caused by the air temperature and the relative humidity, that lead to the coefficient $\alpha$ and can be obtained from Table 2 in the DIN ISO 9613-2 paper [45]. This effect is rising in significance with the distance the wave has traveled and this is represented by the distance value $d$ in the formula.

$$A_{\mathrm{atm}} = \frac{\alpha d}{1000} \tag{6}$$

The ground attenuation $A_{\mathrm{gr}}$ is caused as a result of the reflection of some sound waves from the ground and their interference with the sound propagation between the source

point and the receiver. The effects of this factor are calculated as the sum of the ground attenuation generated in the area near the receiver ($A_\mathrm{r}$), near the source point ($A_\mathrm{s}$) and the area in between the other two ($A_\mathrm{m}$), which is shown in Equation 7. The attenuation for the three areas is decided by the hardness level of the ground material in the respective region. The formulas for calculating $A_\mathrm{s}$, $A_\mathrm{m}$ and $A_\mathrm{r}$ for each octave band frequency can be referenced from Table 3 in the DIN ISO 9613-2 paper [45].

$$A_\mathrm{gr} = A_\mathrm{s} + A_\mathrm{r} + A_\mathrm{m} \tag{7}$$

The attenuation due to shielding $A_\mathrm{bar}$ is created when the sound waves come into contact with objects, which can be classified as barriers based on their size, shape and positioning, and get dampened. The distortion happens both when the sound comes into contact with the vertical side of the object and when it glides in close proximity to its upper horizontal side, these two cases get calculated using Equation 8.

$$A_\mathrm{bar} = \begin{cases} D_\mathrm{z} + A_\mathrm{gr} > 0 & \text{upper edge} \\ D_\mathrm{z} > 0 & \text{else} \end{cases} \tag{8}$$

The shielding dimension $D_z$ for each one of the barriers edges gets calculated using the following Equation 9. It takes into account the effect of ground reflections $C_2$, which in the majority of cases equals 20 dB. Further on this value is multiplied with the $C_3$ value calculated in relation to the wave length $\lambda$, for each of the octave band frequency centers and the distance between the two diffraction points $e$, in case of double diffraction like the one shown in Figure 4. The value $z$ refers to the measure of difference between the path lengths of the diffracted and the direct sound. Finally the value $K_\mathrm{met}$ is a correction factor for meteorological influences.

$$D_\mathrm{z} = 10 \log \left( 3 + \frac{C_2 C_3 z K_\mathrm{met}}{\lambda} \right) \tag{9}$$

More over the factor $C_3$ is calculated by the following Formula 10, related to the wave length and the distance between the two diffraction points.

$$C_3 = \begin{cases} 1 & \text{single diffraction} \\ \frac{1+(\frac{5\lambda}{e})^2}{\frac{1}{3}+(\frac{5\lambda}{e})^2} & \text{else} \end{cases} \tag{10}$$

In the case of a simple diffraction the $z$ index would be calculated by the following Formula 11. Where $d_{ss}$ is the distance between the source point and the first diffraction point, $d_{sr}$ is the distance between the second diffraction point and the receiver point and $a$ is the distance component parallel to the shield edge between source and receptor point, in meters.

$$z = \left( (d_\mathrm{ss} + d_\mathrm{sr} + e)^2 + a^2 \right)^{\frac{1}{2}} - d \tag{11}$$

Finally the formula that brings us the results for the meteorological correction $K_\mathrm{met}$ is shown in Equation 12

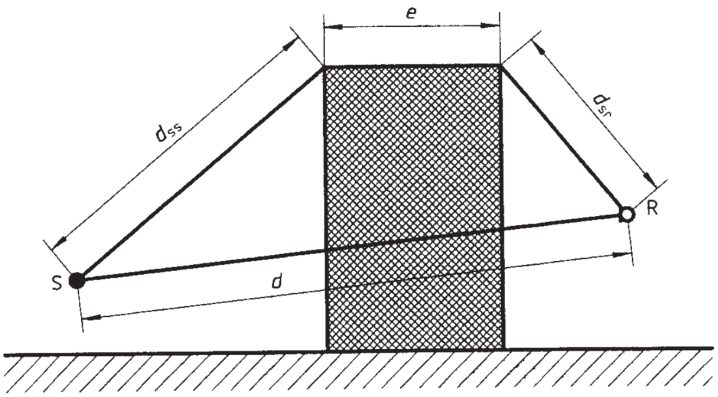



Figure 4: Geometric conditions for determining the shielding value, where $d$ is the distance of the connecting line SR between the source and receiver point, $d_{ss}$ the distance from the source to the (first) diffraction edge, $d_{sr}$ the distance from the (second) diffraction edge to the receiver and $e$ the distance between both diffraction edges [45].

$$K_{\text{met}} = \begin{cases} \exp\left(-\frac{1}{2000}\sqrt{\frac{d_{ss}d_{sr}d}{2z}}\right) & z > 0 \text{ and upper edge} \\ 1 & \text{else} \end{cases} \tag{12}$$

## 3.2 Adjustments in the case of wind turbines

Normally the DIN ISO 9613-2 model is used for cases, where the sound source point is close to the ground level (under 30 meters). Because this is not the case by wind turbines, which are normally much taller than 30 meters, we have extended the DIN ISO 9613-2 model with the Interims procedure (Interimsverfahren) [71], developed for the cases of taller sound sources. This procedure allows us to take some assumptions about some of the normally variable values. Such are the ground attenuation $A_{\text{gr}}$, which we can assume to be equal to -3 dB, the directional correction factor $D_c$ is taken as 0 dB and finally the meteorological correction $C_{\text{met}}$ would also be 0 dB.

## 3.3 Assumptions for our algorithm

Because of performance reasons we have taken some further assumptions, to reduce the runtime of the algorithm. We are not considering barrier objects of any form and measures, but assume that all objects in our calculations have a quadratic base with the width and depth (x,z) of the object equal to 10 meters and variable heights (y). Further on, because of the scale of our approximations, the values for the attenuation factor $A_{\text{misc}}$ caused by various effects, had negligible impact on the simulation, and for that reason it was no longer considered.

| Test number | Implemented | Deviation [dB] |
|---|---|---|
| 01 | Yes | 0 |
| 02 | Yes | 0 |
| 03 | Yes | 0 |
| 04 | Yes | 0,02 |
| 05 | No, because alternative method is not used VDI 4104 | - |
| 06 | Yes | 0,02 |
| 07 | No, because alternative method is not used VDI 4104 | - |
| 08 | Yes | 0,05 |
| 09 | Yes | 0,02 |
| 10 | Yes | 0,05 |
| 11 | Yes | 0,1 |
| 12 | Yes | 0,1 |
| 13 | No, because only cubic buildings are considered | - |
| 14 | No, because only cubic buildings are considered | - |
| 15 | No, because only cubic buildings are considered | - |
| 16 | No, because only cubic buildings are considered | - |
| 17 | No, because only cubic buildings are considered | - |
| 18 | No, because only cubic buildings are considered | - |
| 19 | No, because only cubic buildings are considered | - |

Table 2: Test results according to DIN ISO/TR 17534-3 test cases.

## 3.4 Quality assurance of the model (test from DIN ISO/TR 17534-3)

The DIN ISO/TR 17534-3 paper [69] describes an array of 19 test cases in total, with included input values, as well as expected output values. We have implemented these tests, based on our assumptions and needs and have presented our results in Table 2. Some of the test cases were not suitable with the assumptions we had made and were not considered (13 - 19). The implemented by DIN ISO/TR 17534-3 tests normally are considered as successfully passed if the produced by the model value for the sound power level is with deviation within 0,05 dB from the expected value. Some of our tests have produced results slightly outside of the acceptance range of 0,005 dB, namely tests 11 and 12. This behavior could be caused by rounding issues of the used external libraries. Overall the model behaves as desired and was further implemented into our algorithm for the approximation of noise propagation around wind farms.

# 4 Frontend Recources

## 4.1 Technologies used

For the frontend part of the application we have used modern technologies, suitable to our needs. In this subsection we are going to present our choices and the argumentation behind them.

### 4.1.1 Goals and needs

1. **Easily accessible** The goal of this work was to develop an application that covers a very wide target of potential users. For this reason it had to be accessible on as many as possible different device types. This includes devices with smaller displays and computing power like smartphones or tablets, for example, and also more powerful machines such as desktop computers or other devices with bigger display surface. This also presents the challenge of many different operating systems, for which our application had to be suited. As a result, a technology was needed, that could allow us to write the codebase in one single language and then adapt the code into suitable machine language for the devices we are targeting.

2. **Easy and fast to develop** Because of the wide target audience we also had to put big priority on making the content of our tool accessible, understandable and engaging. However, the tool also had to be developed by only a small team of students in a limited time period, so there was a need to quickly and easily integrate components, with complex logic but simple, well known and visually appealing design.

3. **Robust data management** Another challenge specific to our project was the many information sources that were needed for the construction of the tool. We not only need data produces by different algorithms, but also data from online APIs (for example weather data), different constant values that might change from one wind park to the next and others. This system had to be also very modular to allow for reusability. Thus we required a way to load, organize, store and manage the data in a robust and efficient way that would not overload the hardware or the network connection of the users device, which in some cases might potentially be very limited.

4. **Easily maintainable** We were also looking for a modern technology for the development of the application, in order to ensure that our dependencies are going to be maintained in the future and would not lead to problems. Another advantage of this was that modern technologies are well documented and have large communities behind them, which eases the development process.

5. **Compatibility with already developed tools** Some parts of the tool were already developed or in development at the time this work was initiated, so a technology had to be chosen, which complies with the already developed software. The 3D visualisation component, for example, needed a web based environment to run properly.

### 4.1.2 Hybrid Application (Ionic)

**What is a hybrid application** There are many alternative ways for building a mobile application, but for the development of our project we have chosen to create a so called Hybrid Application [9] with Ionic. This type of application is created using web development technologies like HTML, CSS and JavaScript, but is afterwards encapsulated in a native application wrapper, which gives access to a wide range of capabilities of the user's mobile device. Creating our tool in this manner has a few key advantages, which also solve our needs well.

Firstly the application can be developed and maintained with a single codebase. This means that we could simultaneously build our tool both for web browsers and native mobile applications on both Android and iOS devices. This complies very well with the needs of our small team and limited timeline. Another big advantage, that Ionic has for other hybrid application building technologies, was that it uses web based technologies as the core for the software construction. This allowed us to directly reuse and integrate already developed features like for example the three.js based 3D simulation of real locations [67]. Using web technologies also had a priority for us, because of our previous experience and knowledge with them, which lowered the learning curve and allowed us to quickly advance into building the tool in practice.

Secondly a hybrid application in comparison to a web application would give us the opportunity to use fully, or at least to a large extent, the capabilities of the devices we are targeting. This means for example that we have the option to extract GPS locations, automating our simulation calculation process. Another opportunity is to access the camera or NFC scanning capabilities of the device, which could be used to build an augmented reality alternative to our current fully virtual approach to the simulation. Such variety of options would ensure that future extensions to the tool would be possible and would not be limited by the features of mobile browsers, such as if we had chosen to build a solely web based application.

A third reason for our choice was the option to use native-like UI components to ensure the outlook of our application would stay consistent and visually appealing across all target mediums and devices. Using such native looking components would make the experience more appealing, more intuitive and less confusing for users, that are used to the outlook of their particular operating system [5].

**Alternatives to Hybrid Application** Here we are going to take a look at the different alternative ways to build a mobile application and the reasons which have influenced our choice against them.

- **Native Application** - The most popular mobile operating systems - Android and iOS, both present us with the option to build an application using their native development language and tools. For iOS devices the native applications are built either with Swift [21] or Objective-C [16] and afterwards compiled via XCode [27], while for Android native application the code has to be written ether in Kotlin or Java and compiled using Android Studio [53]. This means that only for our target audiences using mobile devices, we have to develop and maintain two completely

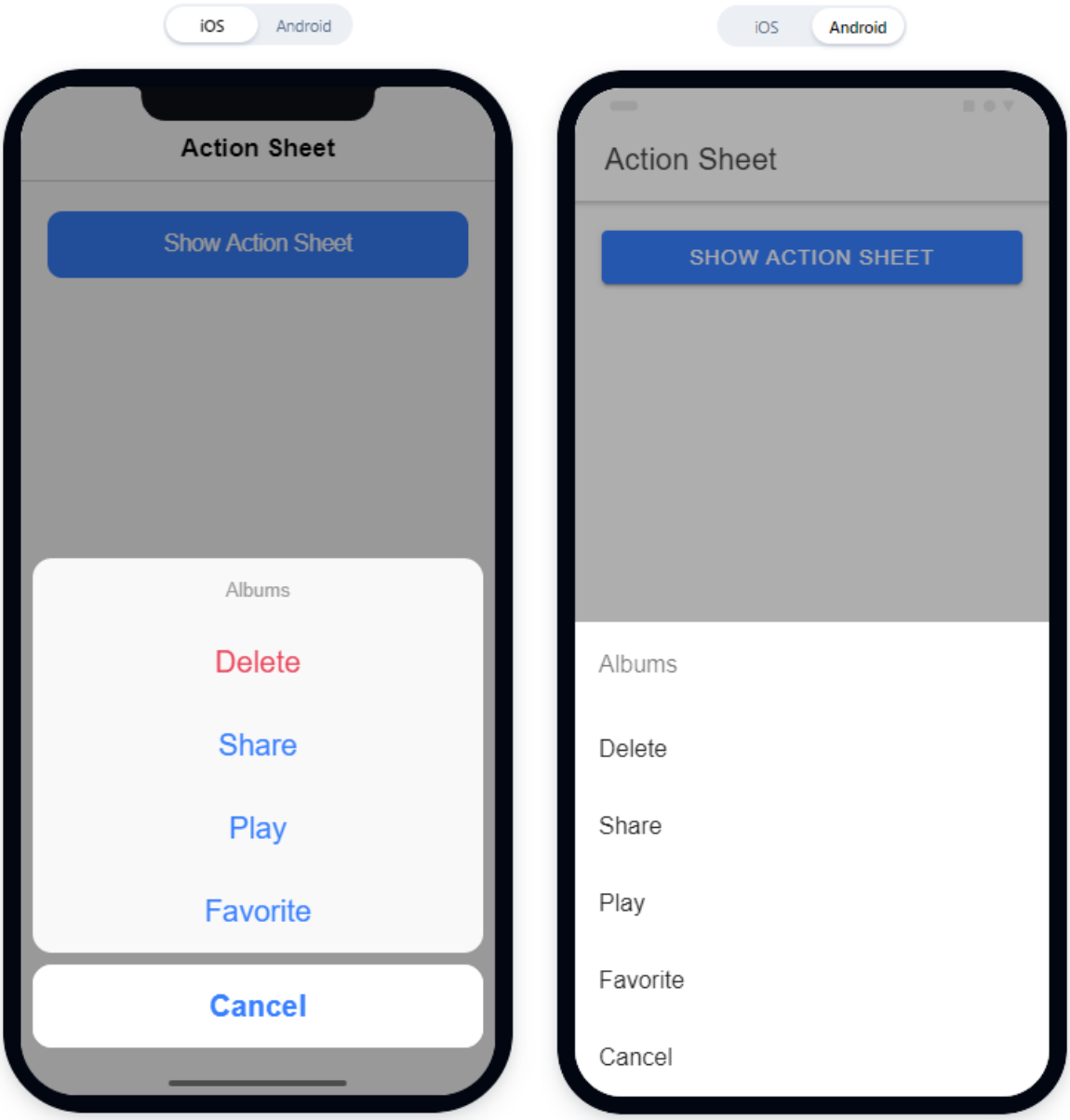



Figure 5: Example of differences in the outlook of a Ionic native-like UI component between iOS (left) and Android (right) [22].

different codebases, each requiring different knowledge and skills to build. This also would not produce a desktop version of the application, whereas with the hybrid application we can deploy our tool as web application as well and thus make it accessible through the browser both on mobile and desktop devices. However, these native applications have the best possible performance, because of their specific builds towards the specific systems. In order to benefit from this better performance, some technologies for building hybrid applications use just separate native UI components, such technologies were, however, also not suitable to our needs (look at point about Hybrid-Native Application).

- **Web Application** - This solution would have suited our need for the tool to be accessible from all platforms, however it would limit the entry to the different sensors and tools of the user's devices. The application would be usable only through a browser so one has to comply with all the requirements of the different browsers, both for their mobile and desktop versions. Another problem with this approach would be that one needs internet connection to view the application, where as with the other approaches we can make parts of the application accessible even offline.

- **PWA (Progressive Web App)** - This option builds onto the previous one and the final product is a web application which has extended features, compared to a normal web application. One such feature is that one can make the application available offline. The first time the user accesses the application through their browser they would be prompted to decide if they would like to add an icon for the application in their mobile menu. After that they can directly load the tool without the explicit need of internet connection. Of course a connection is needed for some of the components of the application to pull resources from external sources, but these can store the latest state of the data, they had received when the application was lastly used with connection to the internet, and display it. We as developers would again, like with the native applications, get to integrate different capabilities of the devices directly, overcoming the restrictions of a browser.

  A disadvantage here is that we would not have direct access to native-looking UI components and although there are different options that offer components with modern and responsive design, these would not be adapted to what the user is normally used to seeing and using on their system. Moreover we would need to always test and adjust the outlook of our application on all browsers and device screen sizes.

- **Cross-platform/Hybrid-Native Application**- There are two general types of technologies for building hybrid applications [4]. The first type, under which Ionic falls, is using web development technologies to build the app structure and the UI components. This code is then embedded into a native application wrapper, which establishes the communication between the application and the device's operation system. This is sometimes referred to as hybrid-web application. The second approach also aims to develop applications in one language and later on compile them into native code for the different mobile devices. However the architecture of this

type of hybrid application is completely different from the web based one. In order to access and use different native UI components, this type of hybrid application allows developers to map specific calls written in the technologies' language to the native UI components of the targeted operating system. For example a text field would be converted into [TextView] for Android and [UiView] for iOS respectively.

This is why this type of hybrid application is sometimes referred to as hybrid-native. Most technologies for building hybrid applications fall into this category, some of the more popular are React Native [18], Xamarin [26], NativeScript [12], Flutter [7]. This approach was considered and even tested for our project however some key disadvantages of it proved to be not suitable. The problem mainly was that in order to integrate the already developed parts of our tool, which were web based, into such native UI components we had to adapt some of them to behave like web browser windows. This means that we would not have taken advantage of the better performance of the native components and thus has proven to be less suitable then the hybrid-web option.

### 4.1.3 Frontend Framework (Angular)

Because Ionic uses web based technologies to build the application, it is possible, although not necessary, to use a JavaScript framework [52]. Although a framework introduces a learning curve and potentially implements some parts that would not be fully utilized, thus increasing the tools storage needs, those inconveniences are majorly overweight by the advantages of such technologies [24]. A framework presents the developer with a already built structure to the application, which eases the workflow, speeds up the development and improves the robustness of the application. If we had to build the whole tool without the help of a JavaScript framework we would not have had access to a way to duplicate and reuse components of our application easily and consistently. Other useful features are the Lifecycle hooks which allow for functionalities to be executed on specific timestamps of a component's attachment to the webpage, updating its related data etc., also a Router which eases the building of complex routing between the pages of our tool and others.

There are three viable options when it comes to JavaScript frameworks. Those are React.js [17], Angular [1] and Vue.js [23]. The choice between them is not an easy one and we have taken a lot of factors into consideration. Some of those are the popularity of the framework under the developers world wide, the learning curve, which has to be overcome before one can start working and using all of these tools features, and lastly the performance of the framework, because especially in our case where large amounts of data have to be loaded and managed, even small performance issues might lead to big load times. Although the results in all those three categories are not in favor of Angular, although not by a large margin [51, 66], we have decided to use it for this project, because of a few advantages it presents.

Firstly Angular implements some patterns that are well known for most computer science students, of which our team consisted. An Angular application is written in TypeScript and then compiled into JavaScript at runtime. This allows for the use of object oriented programming paradigms and rules for the construction of complex behav-

iors. A codebase written in such a way is more easily recognizable and understood for people with experience in the field. TypeScript can also be added to a React.js or Vue.js project, but Angular is built with it in mind and uses its features fully. Another recognizable pattern that Angular is based on is the MVC (Model-View-Controller) architecture [60]. Understanding the overall structure and data flow of the application makes it more appealing and easier to debug for experienced people.

Another argument for Angular is that it is very well suited for managing different data objects, because of its strong typing nature. This means that the codebase would be much more robust and reliable. This prevents problems caused by wrong typing in the data objects very early on and thus is much less likely to introduce hidden errors.

Finally the previous experience of our particular team members was taken into consideration. Some were already familiar with Angular, which eliminated the learning curve and allowed for faster development of functioning software.

### 4.1.4 Data management (NgRx)

One of our above mentioned needs was to have a robust data management structure, to take care of our many and diverse information sources. This is in part already accomplished by TypeScript and its object oriented nature, because we can easily define data models and classes in our Angular application, which would check the structure of the incoming data objects and apply strict typing to their attributes. However we have some frontend components, that require for data from several sources to be requested, adapted for the particular components, stored and fully present at the time these components have to be attached to the visual interface. An example for that could be one of the bar chart components we have planned to include, which would be used to show $CO_2$ currently both for the entire country and for the particular region, where the wind farm is going to take place, and also an approximate prediction of what the levels of $CO_2$ emissions would be after the building of each different scenario for the wind farm of interest. This means that we have data from online sources which show the current $CO_2$ emission levels, and from our own backend source, where the approximation for the emissions, after the wind farm is functional, would be calculated (this component could not be integrated fully in the scope of this work). These sources might(and often do) respond with data objects of different structure, which means that we have to adapt them and ensure that the whole information was fully loaded before transferring it further to the visual representation of the component. Moreover the information we have loaded to the frontend is often required for the rendering of several components and thus has to stay consistent between all such pages and components. By changing the selected scenario, for example, we would have to update all, related with this information, components to display the data for the newly selected wind farm variant.

To take care of this problem we take advantage of a global state management tool, in our case Angular is best compatible with NgRx [14], and in particular its State functionalities. This tool allows us to implement an abstracted Store service, where our global data would be stored and managed. The data inside our Store component is fully isolated from our UI components and can only be changed via special functions called Actions. After triggering an action function the store can get updated via Reducers. If a service has to be used

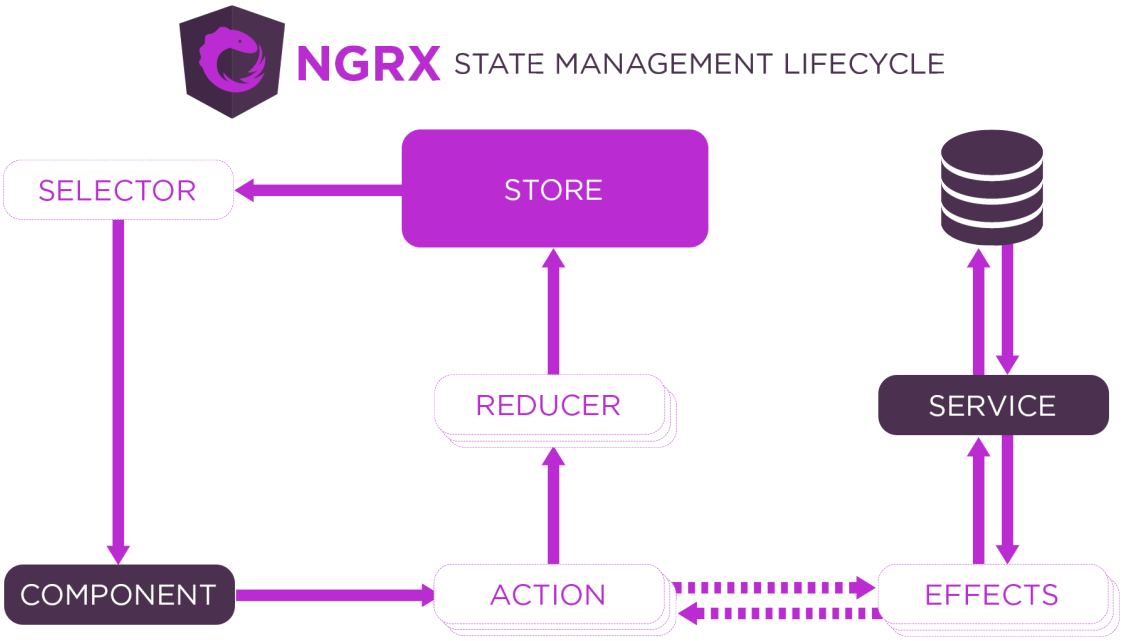



Figure 6: Schema of the state lifecycle in a NgRx Store[14].

to retrieve or rework some of the data, for example by using an API-Service, we can do that via separate special, asynchronously ran functions called Effects. The data from the state can be integrated into the UI components using selectors, which efficiently store the lastly loaded value until a new one is present, this means that the state is not accessed every time a new components needs data from it, but rather only when the values in the state itself change. A schema of the whole work-cycle of the tool is shown in 6

This architecture has some very big advantages towards managing data safely and efficiently. Some of those are [14]:

- **Type Safety** - the use of TypeScript and strictly defined models for our data objects, ensures more robust and cleaner code.
- **Immutability and Performance** - because of the used data structure and the overall architecture, which connects the different building blocks of our store, detecting change and respectively updating the visual interface dynamically on data loads is made easier.
- **Encapsulation** - all the business logic and service calls would be isolated from the UI components and in this manner side effects caused by these functionalities would be limited.
- **Serializability** - the use of observables ensures that the state is predictably stored.
- **Easier Testing and Debugging** - NgRx/Store allows for the use of the StoreDev-Tools, which make testing and debugging of the state and its functions very easy and straightforward.

In the next subsection we will show how we use NgRx in combination with our data sources to connect and manage our frontend components.

## 4.2 Architecture and Connections

Here we are going to present the general structure of the application and show the data flow of information. We will see how the information is created, stored and accessed when needed.

### 4.2.1 Overall architecture

Firstly we are going to look at the overall architecture of the whole tool and its connections, take a look at Figure 7 for a visual representation. We can divide the architecture into three abstract levels, namely:

1. Frontend - responsible for the visual representation of the tool.
2. Backend - responsible for the connection between the different levels and retrieving and managing the data requested by them.
3. Offline calculations - these include all the algorithms that we have which calculate the optimal positioning of the wind turbines, their aural impact, their shadow impact and so on. The information required by those algorithms and also the outputted results from them get stored in a data base, from which they can be retrieved later on.

These levels further on feature more detailed parts. On the frontend we have 2 major data flow paths. For the majority of the data we load the needed information from the data base by requesting it from the backend REST API [19] server, then adapt the received data into suitable data objects and store them into our global NgRx Store. Thereafter these data from the store can be requested via the NgRx selectors by our UI components in order to present it in a visual manner.

However for the 3D simulation component we have implemented a way to export the 3D three.js scene object from the backend. This is done on the following manner. When the user opens the simulation page, they are greeted with a map around the location of the selected wind park scenario. Once they have selected a region they are interested in viewing on the 3D simulation they have to click a button which triggers an API request to the backend and includes the coordinates of all 4 borders of the selected region (north, south, east and west). The backend is programmed to store the lastly requested regions as a 3D scene so it tries to respond as quickly as possible with such pre-generated scene, which is ready to display. If a pre-generated scene for the requested border coordinates is not found the scene is generated automatically, this step can require different amounts of time depending on the size of the scene the user has requested and the density of buildings and other objects in the area. When scene is ready or found in the pre-generated samples, it is exported as GLTF object [8]. This allows us to send it directly via a HTTP response to the frontend. There the GLTF object gets imported and translated into a

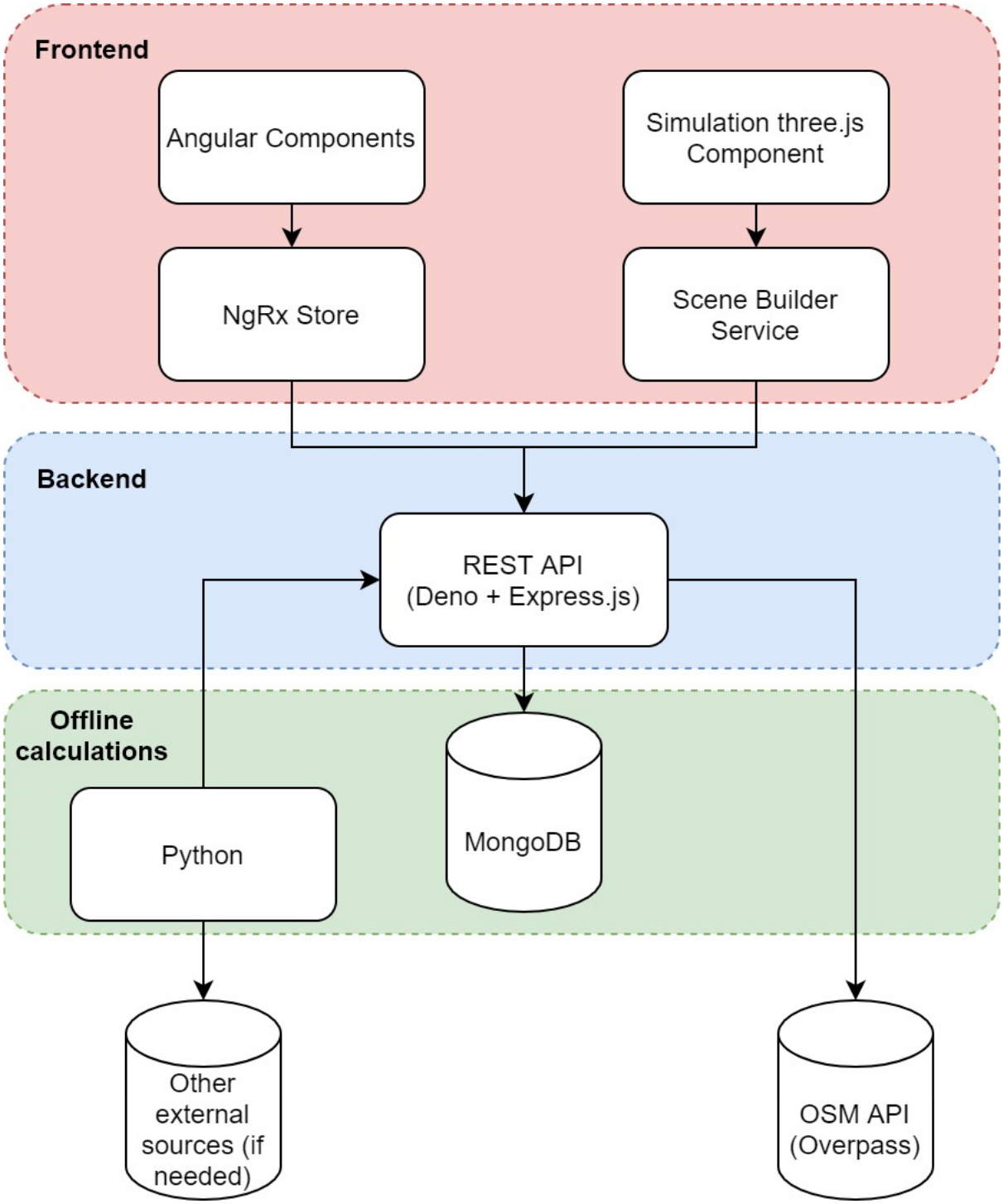



Figure 7: Overall architecture of our tool.

three.js object by a special scene-builder service, implemented by us for this purpose. Once the scene is ready to render it is transferred to the simulation components which displays it on the visual interface of our tool.

The backend server is not only responsible for the communication with the frontend by implementing a REST API structure, but it also implements the functionalities for generating and exporting the 3D scenes for the visual simulation. Because the three.js scene generator needs libraries, which are normally aimed for frontend parts of an application, this functionality could not be implemented using Node.js [15], that is why the backend part of our tool is developed using Deno [5], which is a newer alternative of Node, developed by the same creator, which loads the required libraries from online images of them, rather than storing and installing these locally.

The offline algorithms are planned to calculate the noise propagation, optimized placement of the wind turbines, potential area and its surface and shadow durations in different positions around the wind turbines. The data outputted by those algorithms gets stored in a MongoDb data base [11].

The information needed for the calculation of our noise propagation algorithm is also accessed from this data base. Outputted information is stored there as well.

### 4.2.2 Frontend Store to Backend Connections

Because of the more complex data flow between the global state on the frontend and the data that is sent to the frontend from the backend, we will take a more detailed look at the different data sources and the structure of the global state here.

On the frontend we have 3 general types of data objects.

The first is data related to the theme and styling of the application. This data is planned to stay mostly unchanged however it is modularized so that retheming the tool for new regions and to suit the needs of different clients is made easier.

The second type is data that is specific to the wind farm of interest and the region where it is planned to be build, but does not change under the influence of different scenarios of the wind farm. For example such data could be the current $CO_2$ emission levels for the region where the wind farm would take place.

The third type of data is the one that is changed depending on which scenario the user has chosen to observe. Such data is, for example, the prediction outputs for the noise propagation, the shadow durations and the potential area surface. Each entry of data falling into this category had to be organized into a bigger object with all the relevant information about every scenario.

The goal of our global store component was to keep track of all of the data received to the frontend and update the respective objects. Because we should receive information strictly unrelated to the construction of the visual interface, we have introduced 2 special objects into the global state, which keep track of the contents required for special text-based components and chart-components respectively. These objects combine the wind farm related data with the data of the first above mentioned type, which is related to the theming of the tool, in order to provide all the needed information for the rendering of the components later on. This structure is shown visually on Figure 8.

In order to load up all the needed information into the global state we are using the NgRx Effects, which request the needed information via our API-Service. These special functions allow us to load every piece of our data asynchronously, which improves the performance and load times of the tool and the use of Effects separates the data state from our services, which ensures that any side effects produced by the service functionalities would not affect the information that is currently stored.

## 4.3 UI/UX Design Decisions

The main goal of our tool is to present data for many different aspects of a wind farm in a simple way which would be easy to follow and understand for any user. We strive to show the complex data, that is generated and gathered about a planned wind farm and its environment, in a minimalistic and simple way so that the users would not be overwhelmed, and would not lose interest in the different topics we are trying to cover and inform them on. For this reason we have decided to divide our tool into several tabs, each covering a different topic, important to the environment of a future wind farm. We have designed the application to present the information in concise, visual and interactive manner, allowing the user to not only observe the data, but also change aspects of it and observe the consequences, casted by these changes. Observing these tabs in consecutive manner gives the user a well rounded picture of the situation they would have to find themselves, once a wind farm is built in their region.

After the user is ready to move on onto the next topic they can do so from the carousel-type tabs menu, which is visible at all times in the bottom of the screen. Because we wanted to present data about at least 6 different topics (not counting the tab aimed at prompting the user to give their feedback about their preferences according the wind farm scenarios) and normally such bottom situated menu component is meant for no more than 5 tabs [10], for readability reasons, we have made this menu scrollable. This means that the menu works well and makes all the tabs accessible even on smaller devices with narrower screen resolutions. On wide enough displays with enough room to fit all of the tab icons the menu remains static.

For the selection of different scenarios we have added a selector component above the tabs menu in the lower part of the display. There the user can at all times see the selected scenario label and swap to a different one, to observe the changes it would cause. Because of the limited screen space, we have chosen to make this component hide upon scrolling the page. This means that the user would be able to consume more of the content when browsing through the application. We have chosen to present all the possible scenarios as points along a slider. In this way the user would be presented with all possible choices, which would have not been the case with a dropdown or a selection field, where only the selected option is visible. The slider is designed to snap onto the scenario points for better user experience.

We have chosen to follow a more minimalistic approach towards the design of our components and of the outlook of the tool as a whole. The information is divided into small card-looking containers, each featuring a different information piece. This way, the user can clearly distinguish between the different components and would not get confused or mix up the information from two adjacent components. Such a design is less likely to

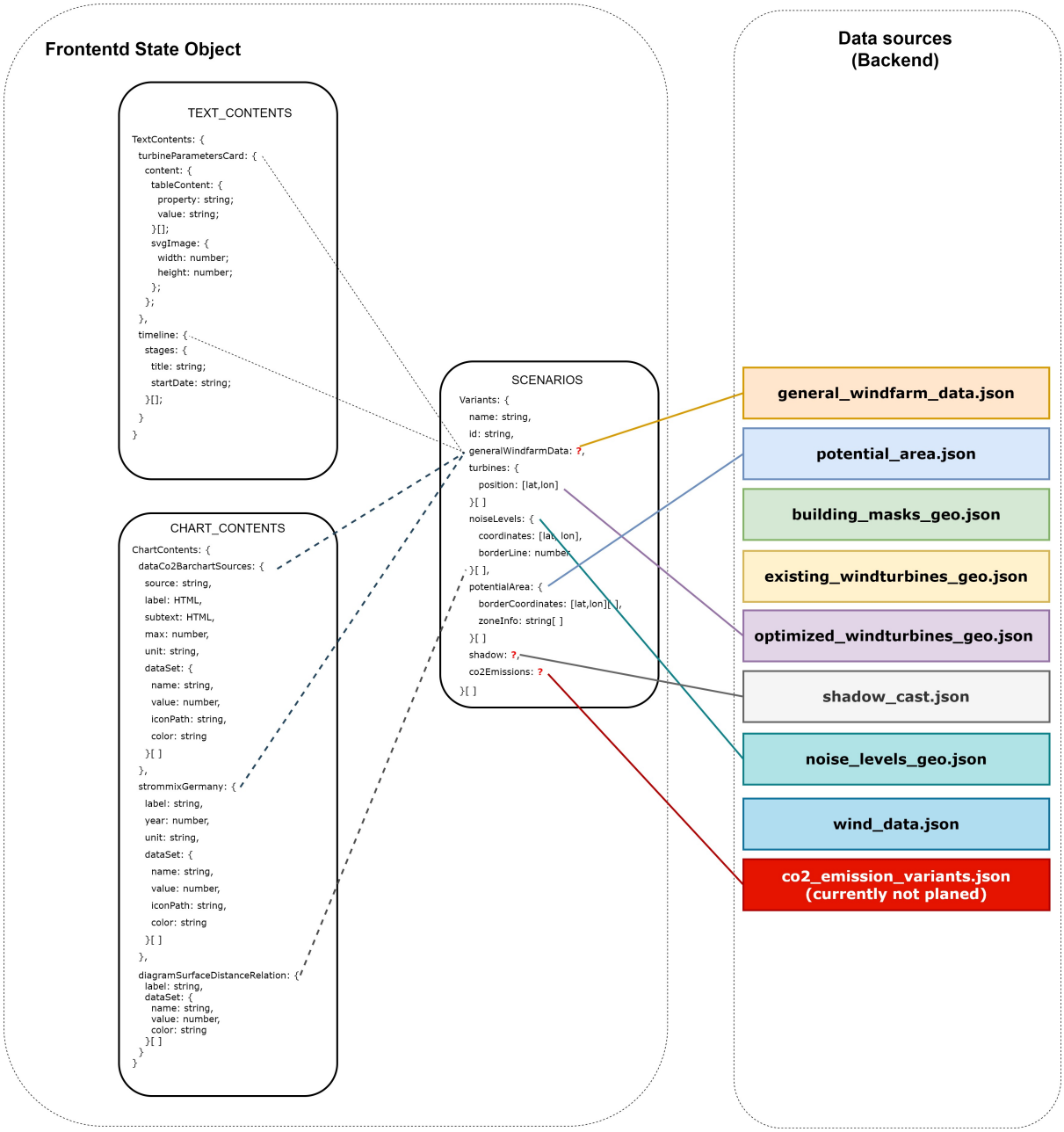
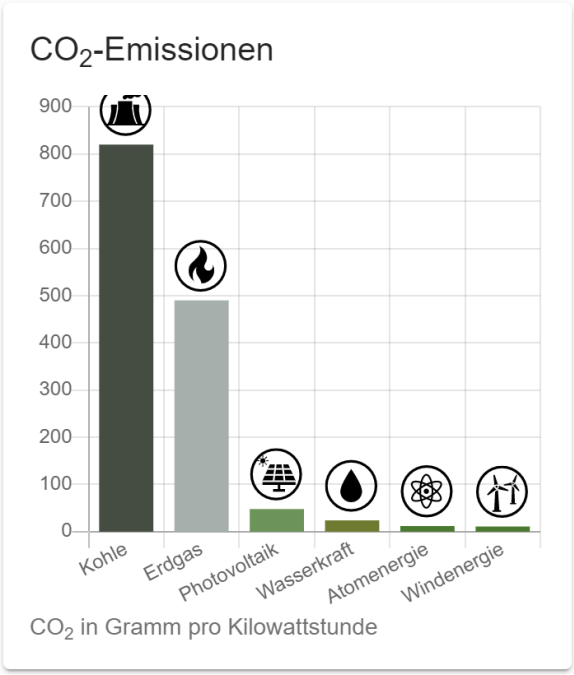
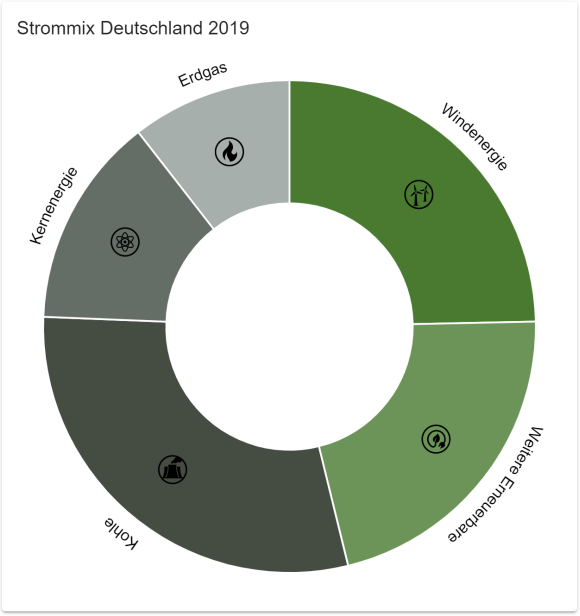



Figure 8: A visual representation of the data flow between backend and frontend global state. Left side shows the global frontend store and its division into three main elements, namely text contents, chart contents and variants information. On the right side we have a color coded list of all the available data files which can be requested from the backend. The lines between the frontend and the backend (solid) represent requests towards the respective data file. The lines between different global state child objects (dashed) represent duplication of the data stored in the respective scenario attribute.




(a) A bar chart showing $CO_2$ emission sources.




(b) A doughnut chart showing the distribution of electricity consumption according to the different sources of its production.

Figure 9: Examples of different charts from our application.

overload or tire the user, thus keeping their attention for longer.

The color palette is also simplistic, featuring very small number of different colors. The theme is easily swappable and adjustable, because we have defined it modularly using Sass variables [20]. The application also implements a dark theme, which is not used at the moment, but is available as a future feature. For the charts and special components, we have used colors, which naturally associate with the respective information they represent. For example in a chart, showing the different energy sources in Germany in the last years, we have used more green-looking colors for the renewable energy sources and gray ones for the others.

For the different diagrams and charts we have integrated into the tool we have used an external library called Charts.js [3] in combination with its Angular directive - ng2-charts [13]. This library is very suitable for our needs, because with it we could quickly and easily integrate interactive charts with dynamic scaling and fluid animations, which point the attention of the user towards the made changes, examples of the results are present in Figure 9. Charts.js also allows for very detailed customizations via its plugin feature. We have used this feature to make our charts even more readable, by placing the labels for the different data entries directly near the data for them, instead of in a separate legend, and even included pictures to them, to make the application more accessible for any users regardless of their previous knowledge in the field of energy production and consumption.

## 4.4 Integration of the Visualization component

A big part of our tool is a 3D simulation which is generated from a specific region the user selects, where the planned wind farm can be made visible. This provides the user with the as close to realistic as possible impressions of how the wind farm would look surrounded by its environment. The user is able to move the camera and observe the virtually generated buildings, roads, trees and others, as well as choose to include in this virtual scene the planned wind farm in every scenario. In this subsection we are going to see how this is done in a more technical detail and see where possible performance issues may be possible.

### 4.4.1 Selecting area of interest

When the user navigates to the visual simulation page, they are greeted with a map of the general region of interest near the planned wind farm. On the map the wind farm is visible at the location stored in the global store. Once the user has selected a region which interests them, they can click a button to initiate the request for a 3D scene generation. The request includes the 4 border coordinates of the selected region on the users screen as query parameters.

### 4.4.2 Generating scene

Once the query is registered on the backend the 3D scene generation process begins. The data we are using for the generation of real life objects, such as buildings, roads and others are taken from the Open Street Map [29] API, called Overpass Turbo [30]. Open Street Map is a large data base of different objects and their real life locations. This data is generated by the users and as a result it presents an opportunity for very accurate data on the objects in more busy areas. It is also possible for local people, living near a wind farm, to contribute as well, thus making our simulation more accurate as a consequences.

Because the data for every object of interest has to be received from the OSM API, this step is prone to generating large wait times, especially if the user has selected a large or very densely built area. For this reason we are storing the generated 3D scenes for the lastly requested areas and in case the user requests one of them we are ready to send it without any waiting time.

After the data about the OSM objects is received it is converted into a suitable format and from the coordinates a 3D scene is generated using the library three.js [32]. Three.js is an open source library which is aimed at presenting the developers with ways to make lightweight, highly customizable 3D scenes, including animated objects of various mesh shapes, as well as options for importing 3D models. This is all achieved with web based tools, so it makes these results possible in every browser or other web environment, which supports JavaScript. The interest of this work is not focused on the process of converting the geolocational data into accurate representation of the real life objects, so we would not delve into more details about this step but more information can be found in [67].

### 4.4.3 Transferring scene to frontend

In order to transfer the scene object via HTTP request we have to convert it into a GLTF format [8], which is suitable for this task. In order to optimize performance and reduce load times the objects in the scene are merged into a single mesh.

On receiving the GLTF object from the backend, our scene builder service is called. The object is translated there back to three.js 3D scene. After this step it is finally passed onto the visual component where it is rendered.

### 4.4.4 Adding virtual objects

In order to load virtual objects, which are not generated from real life geolocational data, like for example the wind turbine models, we can add them to the three.js scene after its generation. The relative distance between the wind farm, the camera and the objects around it is calculated based on the difference in the coordinates of the center point of the map and the coordinates of the point where the wind farm is located, taken from the map where the user selects a region of interest. The loaded wind turbine 3D models are resized and placed according to our scenario data. These models are dynamic and have rotating rotors, the speed of which can be additionally customized further.

The scene also features shadow representation capabilities based on the intensity and position of the light sources, representing the sun in the 3D environment. These variables can also be regulated dynamically.

## 5 Results and Discussion

In order to inform better the local people, living near a location planned for building a wind park, and let them share their opinion, thus bettering the communication with the institutions, behind the project, we have build a platform which is easily accessible and presents the needed information in a simple and most importantly interactive way. This is planned to lower the cases of unsatisfied locals and optimize the potential area for building wind farms in the future.

As mentioned above, we are not only showing complex quantitative data in a more human friendly visual manner, but we have also made this visual representations interactive. One of the main goals for us was to make the differences in separate wind farm scenarios as apparent as possible, so that the users of our tool can easily meet their decision about which variant for the new wind farm would be of least inconvenience to them personally, both visually and aurally, and which would bring the most valuable trade-off between those inconveniences and the produced energy. To achieve this goal we have included a scenario selection component to each one of the pages for our tool, which allows the user to dynamically change the selected scenario and observe how this affects the different aspects of the wind farms environment. This scenario selection is connected with the global state of the application, meaning that the selection would persist between the different pages, thus limiting confusion for the users.

The product of this work is a hybrid application, aimed to be used on various devices, presenting the most relevant aspects of a wind farm construction to the potential

residents, living near an area, considered for the building sight. We have constructed a simplistic interface, which would catch the users attention and would present data, normally accessible only from technical sources, in a concise and easy to understand fashion. The application is designed to be modular and provide with the opportunity to swap the data sources, thus reusing the interface to present different wind farm projects. The data, not connected to a specific project, is pulled from different currently accurate external data bases, like OpenStreetMap [29].

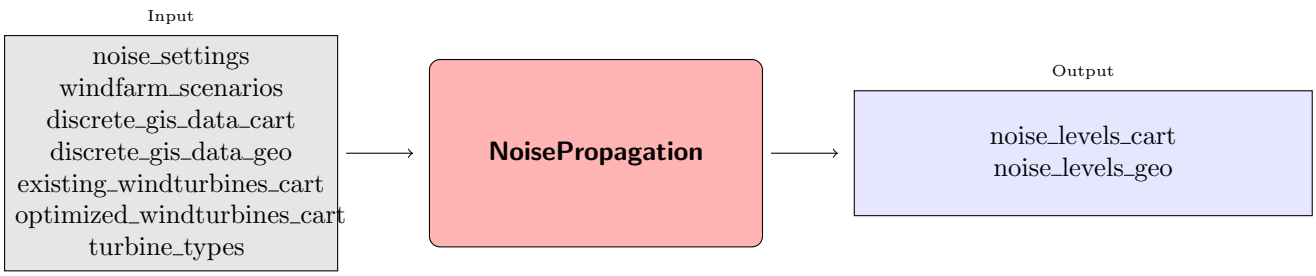
## 5.1 Noise Propagation Algorithm

One of the approximations that we had to compute was the noise propagation of the sound waves, generated by the wind turbines in the planned wind farm. We had to take into consideration the obstacles, which would be present around the wind farm and using the DIN ISO 9613-2 model [45], make a realistic prediction for which areas in the surroundings would be able to sense noises above certain predefined thresholds.

Normally the DIN ISO 9613-2 model is aimed to approximate the sound propagation between one source point and one receiver point in relation to at maximum one barrier at a time. In an average case that we have to consider for a realistic wind farm and the area around such, would mean that we would have to make calculations for each of the hundreds of thousands (around a million in our test case) receiver point approximated in relation with each of our turbine (in our test case 3) and each probable barrier (above 1000 in our test case). This calculations would have taken on average more than 6 months to terminate with a usable output. For this reason we have implemented some optimization steps which reduce the run time of the algorithm for our particular test case (3 turbines, 96 directions (relative for an optimization step 3), threshold levels from 25 dB to 45 dB) to about 6 hours.

We are now going to take a look at our implementation of the algorithm in detail. The input data that we get consists of the target borderlines in decibels and a number of directions (relative for step 3 of the algorithm), as well as information from real sources or outputted by another one of our algorithms about the meteorological conditions, the buildings in the area and about the positioning and size of the existing turbines and the turbine locations, suggested by our algorithm for optimizing wind farm surface area. The output of our algorithm lists for every scenario of the wind farm the calculated border point coordinates for every noise border line level.

### Input and output of this method



**Input files**

`noise_settings.json`

- num_directions
  - INTEGER
- target_decibels
  - INTEGER[]

| | |
|---|---|
| `num_directions` | Number of directions used to find the border line coordinates. |
| `target_decibels` | Decibel values for the border lines. |

**Output files**

`noise_levels_cart.json`

- <scenario_id>
  - border_line
    - DOUBLE
  - coordinates
    - DOUBLE[*][2]

| | |
|---|---|
| `border_line` | Decibel value of the border line. |
| `coordinates` | List of Cartesian coordinates defining the polygon border for the border line. |

`noise_levels_geo.json`

- <scenario_id>
  - border_line
    - DOUBLE
  - coordinates
    - DOUBLE[*][2]

| | |
|---|---|
| `border_line` | Decibel value of the border line. |
| `coordinates` | List of geographic coordinates defining the polygon border for the border line. |

Our algorithm is divided into 3 general steps. In the first one we filter out the number of relevant objects, that we have to consider. In the second one we evaluate the noise levels at each unfiltered receiver point. Finally, in the third step we find out the receiver points, located at the border lines of each area with noise levels above a certain threshold from the ones given as input to the algorithm.

### 5.1.1 Step 1 (Filtering input data)

Firstly we have to filter out the irrelevant objects from the input data, as they would not affect the results. We do this by calculating the maximal range a noise with power level,

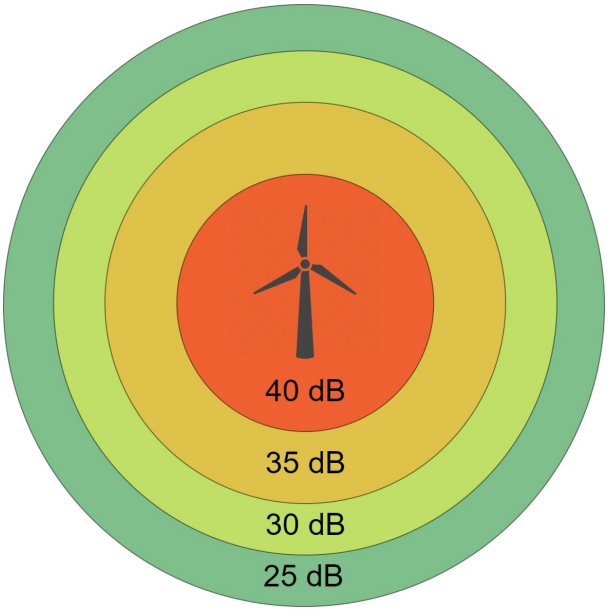



Figure 10: Example of ranges of different areas, where a sound above each one of the threshold noise levels we get as input, is estimated to be measured.

above each of the thresholds from our input, can travel. Then we construct doughnut shaped areas around both the real wind turbines which are currently in the area, as well as the ones outputted by our optimization algorithm. We filter out all receiver points and buildings from the regions outside of the range for our lowest noise level. An example of these doughnut-shaped regions can be seen on Figure 10.

### 5.1.2 Step 2 (Calculate noise levels at receivers)

In the second step we have to compute the noise levels for every unfiltered receiver point in each doughnut. Here we can optimize our computations again, because we do not have to measure the noise levels for each receiver point with every building considered for potential sound barrier. In reality, especially for our case with an emitter point which is very high above the ground level, the objects which might affect the noise propagation in a significant way are the ones, which are not only between the emitter and the receiver but also in relatively close proximity to the receiver. For this reason we consider as sound shields only such objects, which are in the same doughnut noise level area and at the same time in the same quadrant, received by dividing each doughnut by longitude and latitude into 4 parts. An example for this is shown in Figure 11.

The DIN ISO 9613-2 model can calculate the noise level propagation to a receiver with the presence of a single barrier object in between this receiver and the emitter. This means that we have to calculate the noise propagation for each receiver in relation to each emitter and barrier. However it might be the case that several barriers, of the ones we are considering, affect the noise propagation. In this case we have opted to store the minimal sound power level, received by our calculations. Potentially this might lead to slight inconsistencies in cases when the dampening effects of two or more objects influence the

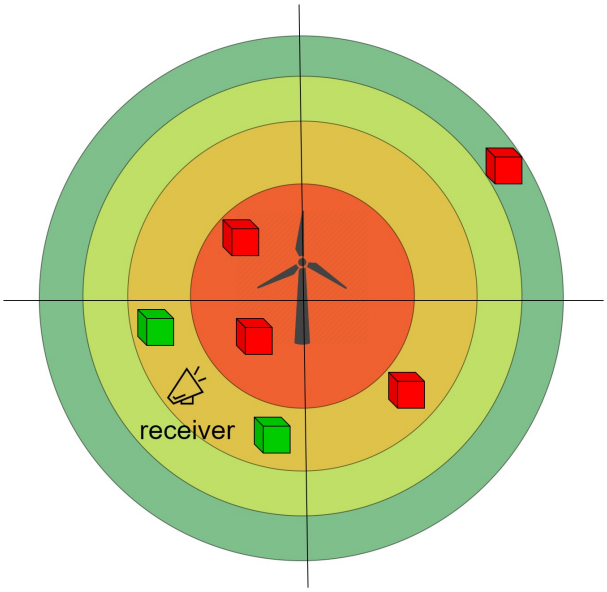



Figure 11: Example of possible distribution of receiver point (marked with an icon) and possible barriers (marked with cubes), colored in green or red depending on weather they would be considered for the respective receiver point or not. We are going to consider in our calculations only the buildings within the second doughnut region in the lower left quadrant, because our receiver point is also located there. In this particular case neither from the approved points would affect the noise propagation between the emitter and receiver, because they are not between them.

noise level at the receiver point, but the differences should not be significant in practice.

### 5.1.3 Step 3 (Determine border lines)

In the third step we are going to limit the outputted positions for each noise level area to the ones, which are located near the border of the area. In this way we are going to output an array of border points for each area, which thereafter would be used on the frontend of our application to present the noise propagation data. Using the border points we can render a polygon, which would enclose each of the areas. Further more it was planned for each polygon to be filled with a semi-transparent color coded background.

In order to optimize the algorithm further we have implemented a way for border line points around all turbines in our wind farm to be calculated simultaneously. We do that as follows. We find the point which is centroid for the wind turbines, both real and planned. Afterwards we divide the whole regional area around the wind farm into equal parts, so that their number equals the number of lines, given as input. Then we iterate along the length of each division line we have created and store each receiver point, where the next encountered receiver point along the same line falls into another noise level range. In this manner we are going to find some of the border line positions of the different noise areas.

This way of calculating the border line points is highly reliant on the number of directions we are going to allow in our input. The higher the number of lines, the more time

consuming the computations become, but the more precise the output is. Because we are going to use these points to build a polygon by connecting them, we do not need a perfectly precise approximation, so it might even be more suitable to save some performance both on the backend calculation and on the rendering of the polygons, by choosing a lower number of directions and as a result outputting a lower number of border points. However, lowering the number of lines too much would not give us precise enough locations, or even none at all. This relation is shown on Figure 12 and Figure 14.

### 5.1.4 Ordering the border line points into polygons

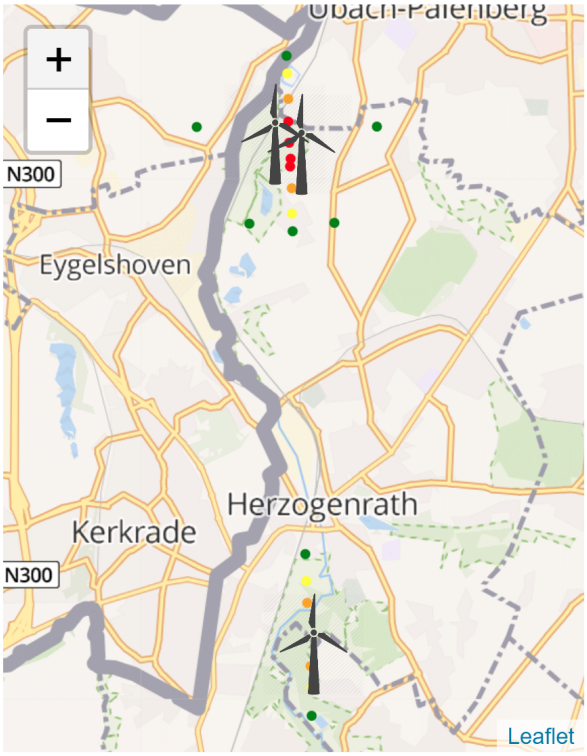
Finally we had to prepare the outputted data for our displaying needs on the frontend part of the application. This is necessary, because of the way we find out and store the border line points for each region in Step 3. Had we taken the coordinate points without ordering them additionally, we would have not been able to show a planar picture just by connecting them incrementally following their indices.

**Graham scan** In order to prepare the outputted data consistently we have implemented a Graham scan [49] step, after the main calculations are of our algorithm are finished. This is done in the following way.

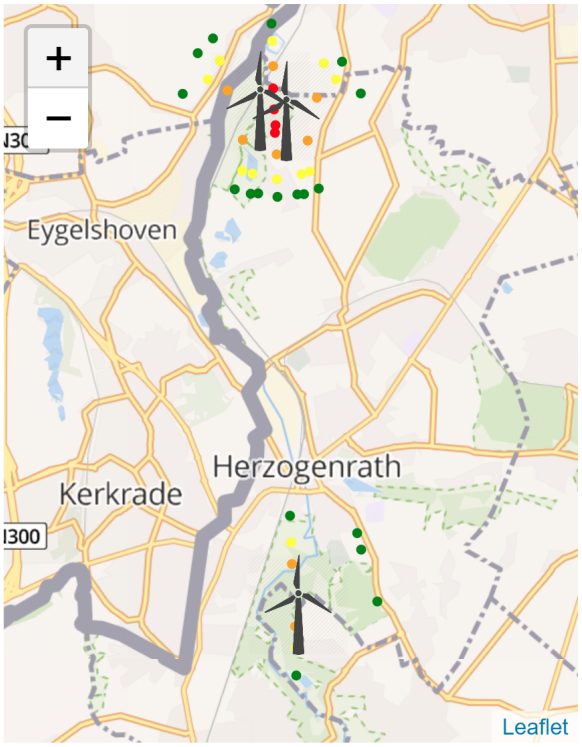
Firstly we need to divide the coordinate points into clusters, because some cases of wind farms have their turbines into smaller clusters which then spread onto a larger area. Wrapping such a wind park into one large polygon would not satisfy our needs and would be a false representation of the data we have calculated. This step is done by iteratively checking the differences between longitude and latitude coordinates of all points and dividing them into separate lists, if the distance is larger than a constant chosen by us.

Then for each cluster of closely plotted points we execute a Graham scan. The idea of this scan is to iterate over the points and accept in counter-clockwise order the ones which are located on the outer edge of the cluster. Firstly we store the point with lowest longitude, or in case that we find multiple such points we take the one with lowest latitude, so we end up with the most west-southern point from the cluster and we store this point as $P$. Then we sort the points in ascending order of the angle that would be created between the vector between the point of interest and $P$ and the latitude-vector ($x$-vector). Then we iterate the sorted points and store points in a stack structure, if they make a counter-clockwise (left) turn relative to the last two points on the stack. If the turn is clockwise (right), then we pop the last entry on the stack. To determine if the vector between the last two points on the stack and the one, between the last and the next, make a clockwise or counter-clockwise angle, we use cross product of the vectors and if we get a positive result, then we know that the angle would be counter-clockwise (left), if the result is negative, we know that the angle would be clockwise (right) respectively. We terminate once we have reached $P$.

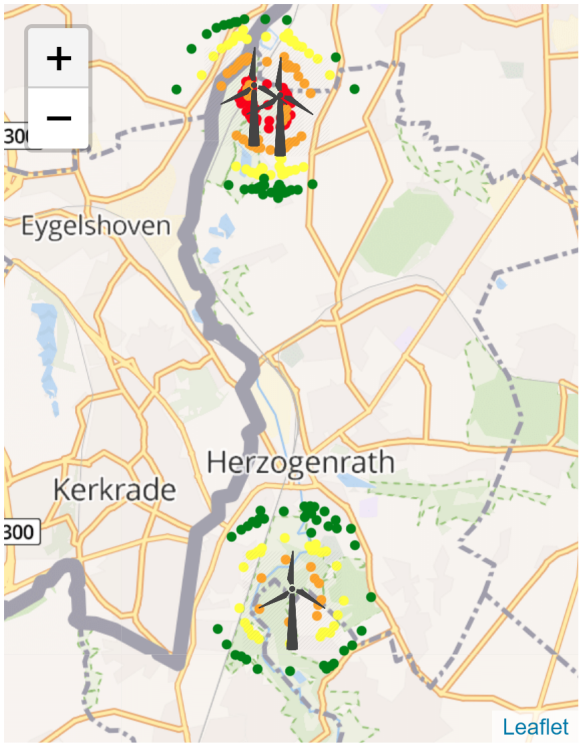
**Time complexity** This scan can be calculated in $\mathcal{O}(n \log n)$ for the amount of points $n$. The sorting of the points takes $\mathcal{O}(n \log n)$ time and the scanning itself takes $\mathcal{O}(n)$, because each point is considered at most 2 times. The clustering part of our computations is done in $\mathcal{O}(n)$, as it loops through every point once.



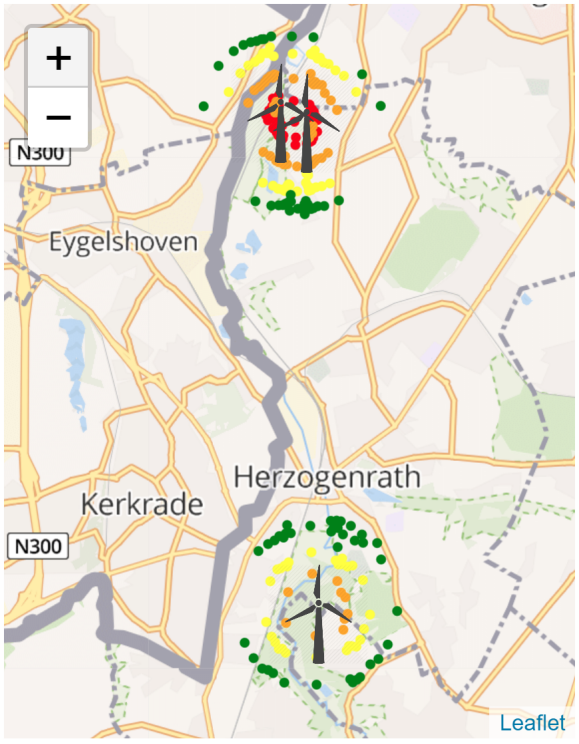

(a) Output for 16 directions




(b) Output for 32 directions

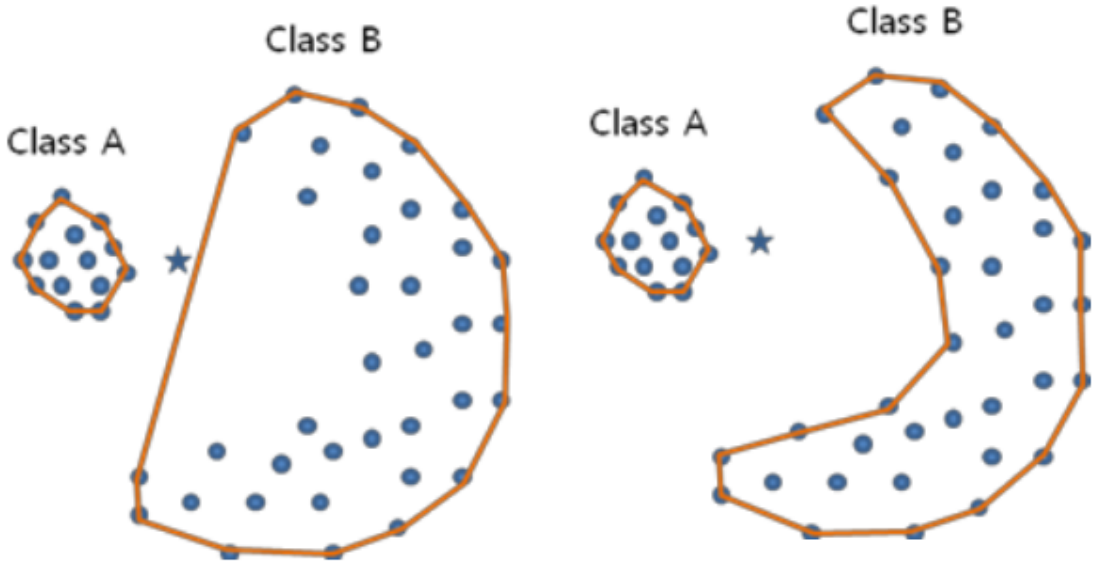



(c) Output for 64 directions




(d) Output for 96 directions

Figure 12: Examples of the relation between the number of directional lines, we are allowing our algorithm to search border line points at, and the precision of the outputted coordinates.




(a) An example of convex hull calculated.

(b) An example of concave hull calculated.

Figure 13: Differences between convex and concave hull computation. Taken from [61].

**Problems with this approach** For our needs this approach could give us approximately accurate results in acceptable times. However it produces a convex hull which encloses some points inside of itself. Example for this behavior is shown on Figure 13. In a perfect scenario a concave hull would be the most accurate solution for us, however a suitable solution for the amount of points, that we potentially would have to consider, was not found in the scope of this work. A proposition for the calculation of a concave hull is given in [61], but the runtimes presented there could potentially lead to big calculation times as the worst case time complexity, which they derive is $\mathcal{O}(n^3)$.

The final results with the additional step which ensures that the border line corner coordinates are ordered into a planar graph is shown in Figure 14.

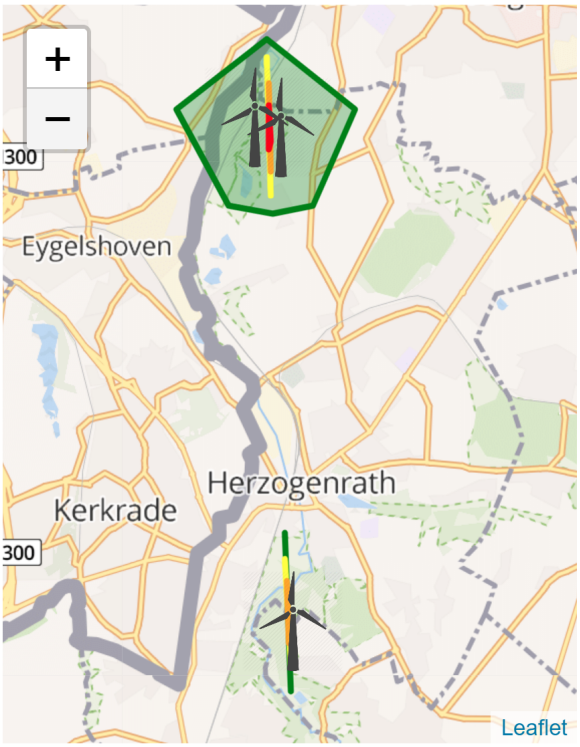
## 5.2 Final outlook of the application

In this section we are going to take a look at the implemented functionalities and outlook of the application done in this work. We are going to look at each of the more important pages of the final application in order to give a perspective to the shown information and view the functionalities implemented there in detail.

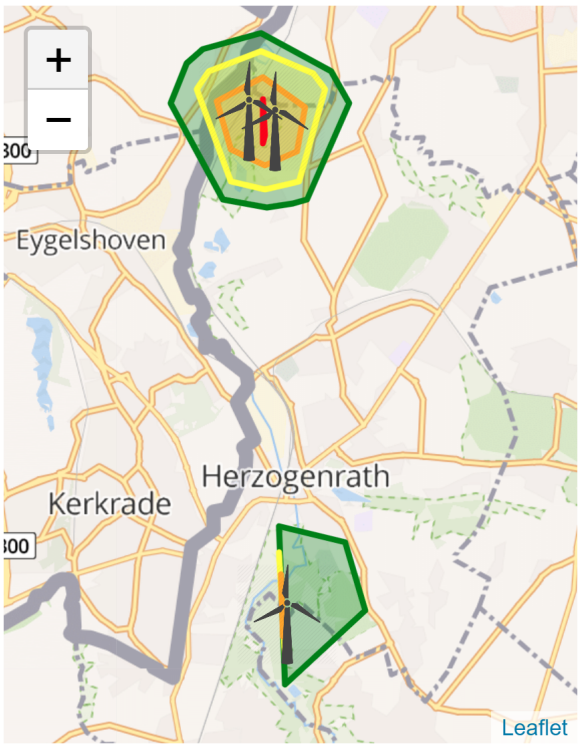
### 5.2.1 General wind farm information page

This is the first page of our application and as such it serves as a greeting of the user. It presents general information about the wind farm and some information related to the use of renewable energy sources.

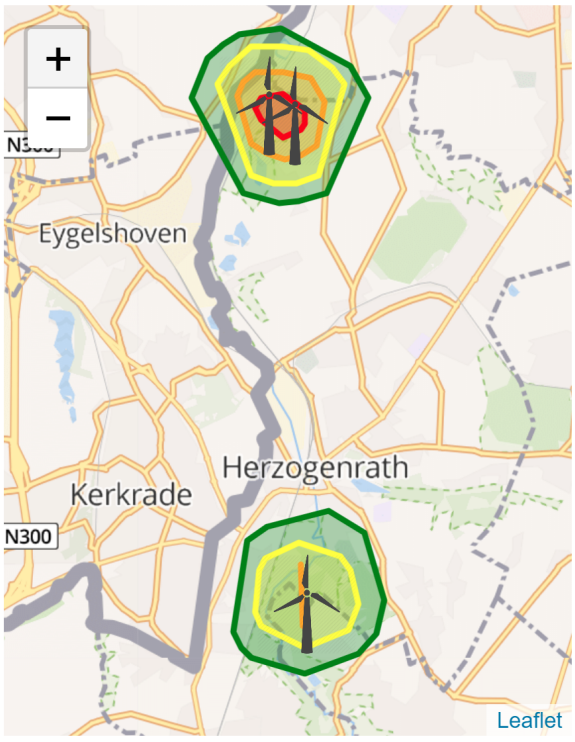
The first card components on the page presents a miniature of a wind turbine, which is animated and is designed to quickly catch the users attention and point them to the



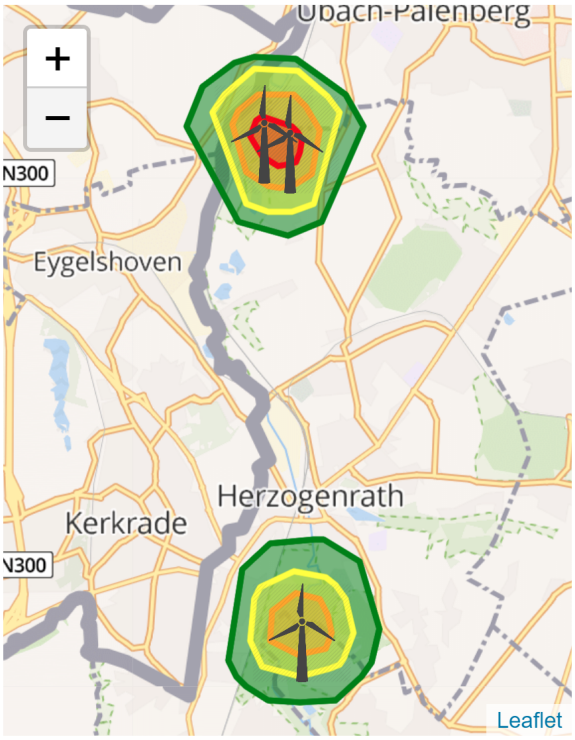

(a) Output for 16 directions




(b) Output for 32 directions




(c) Output for 64 directions




(d) Output for 96 directions

Figure 14: Examples of the relation between the number of directional lines, we are allowing our algorithm to search border line points at, and the precision of the outputted coordinates. Here we have used the data with the additional last step from our noise propagation algorithm and plotted the results as polygons.

relevant information. This graphical element also shows the rotor diameter and the height of the turbine in question. There is also a table with some more specific information about the models, dimensions and productivity potential of the turbines. This table also features an approximation of how many households would be able to be powered using the energy of the selected wind farm scenario. The data shown both on the graphic and in the table is connected to the global store of the application and in turn to the scenario selector component, which is also available on this page, but is not shown in Figure 15 and 16 for visibility reasons. An outlook of this component is presented on Figure 15a.

The second card component shows information about the planned stages of construction for the wind farm construction project. There the user would get an idea of how long the construction process would take and more importantly, when it is expected for the wind farm to be put into production. An outlook of this component is presented on Figure 15b.

The following components on the informational page are meant to give the user a perspective of the current situation with the use of renewable sources for electricity production. The information is presented in the form of interactable charts. In this way the user can quickly compare the entries for each field and get an idea of the general distribution of renewable and traditional electricity sources, both under the produced and used energy. Both charts also feature the option for the separate data entries to be selected in order to find out more detailed information on the particular entries value. An outlook of both of these cards is presented on Figure 16.

### 5.2.2 Noise propagation page

This page shows the results from our noise propagation prediction algorithm. The algorithm outputs a series of different location, divided into different noise level groups. We have chosen to present these results on a map. The user can navigate the map and control the zoom to view the aural influence of the wind farm onto the environment. Each predicted location is shown on the map under the color of its respective noise level. The user can change the chosen scenario to view different versions of the noise propagation predictions.

Furthermore we have implemented two different ways of presenting the results from our noise propagation algorithm. The map can either show the boundary points directly on the map, or in the case that the last step of our noise propagation algorithm was executed and there is suitable information for rendering polygons, then the different sound regions are wrapped inside of color coded polygons. We have not settled on one of the two approaches, because a trade-off has to be made either in favor of precision, with the border line points circles approach, or in favor of more user friendly and easy to understand design.

Results for this page are shown on Figure 17.

### 5.2.3 3D simulation page

To give the residents as accurate idea as possible to the consequences, which a wind farm near their area might pose, we have integrated in our application a 3D simulation,

generated on our backend module [67].

When the page is initiated at first the user is presented with a map of the general location of the planned wind farm scenario they have chosen. They can navigate a map and choose a location they are interested in, once they are ready they can initiate the 3D simulation generation by pressing a button. Once the scene is generated from our backend and received back onto our frontend part of the application, we are presenting it on the graphical interface. The user can then move and rotate the camera in order to view the surrounding objects from different angles. We have also integrated the option for the user to change the chosen scenario and a virtual wind farm under the description of the chosen scenario would be added to the 3D environment. An example of both the map component and the already initiated simulation are shown on Figure 18.

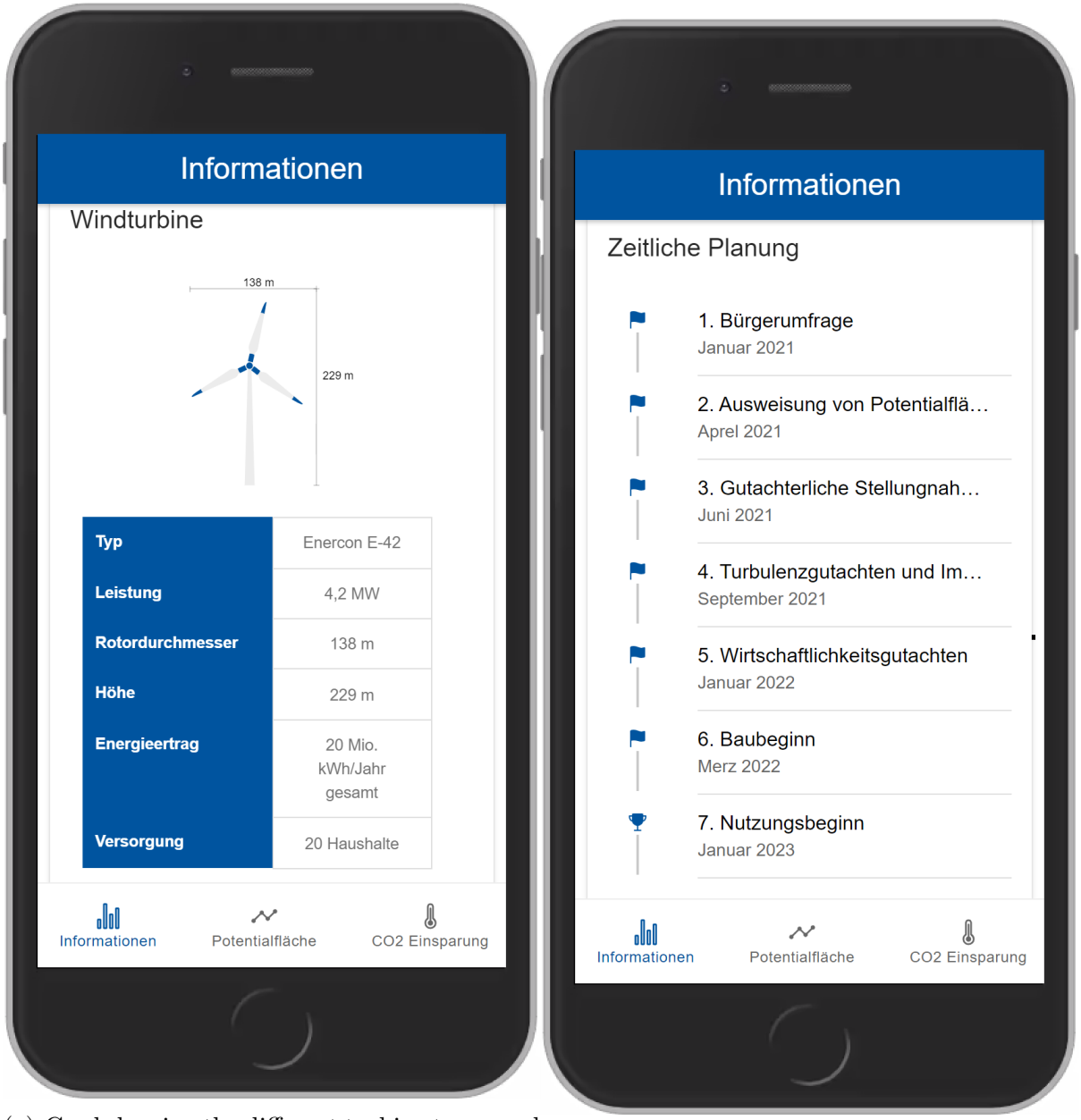
## 5.3 Discussion

The results of our noise propagation approximation algorithm would have to further be transformed from a convex to a concave hull, if we want to display the regions as polygons, but also improve the precision of the final render.

In order to make the application easy to use, avoid confusion and generally improve the users interest in using our application, we need to make our tool responsive, usable on every display resolution, including mobile ones, and let the user know at every time what the state of the application is.

Our application implements an array of design choices and features to accomplish these goals. All of the pages of our application scale according to the display size of the device and would not cause difficulties for the user to see, read or interact with the different components and information we have presented. We have used the provided by Ionic UI components to ensure high standard, appealing design and good performance for our tool.

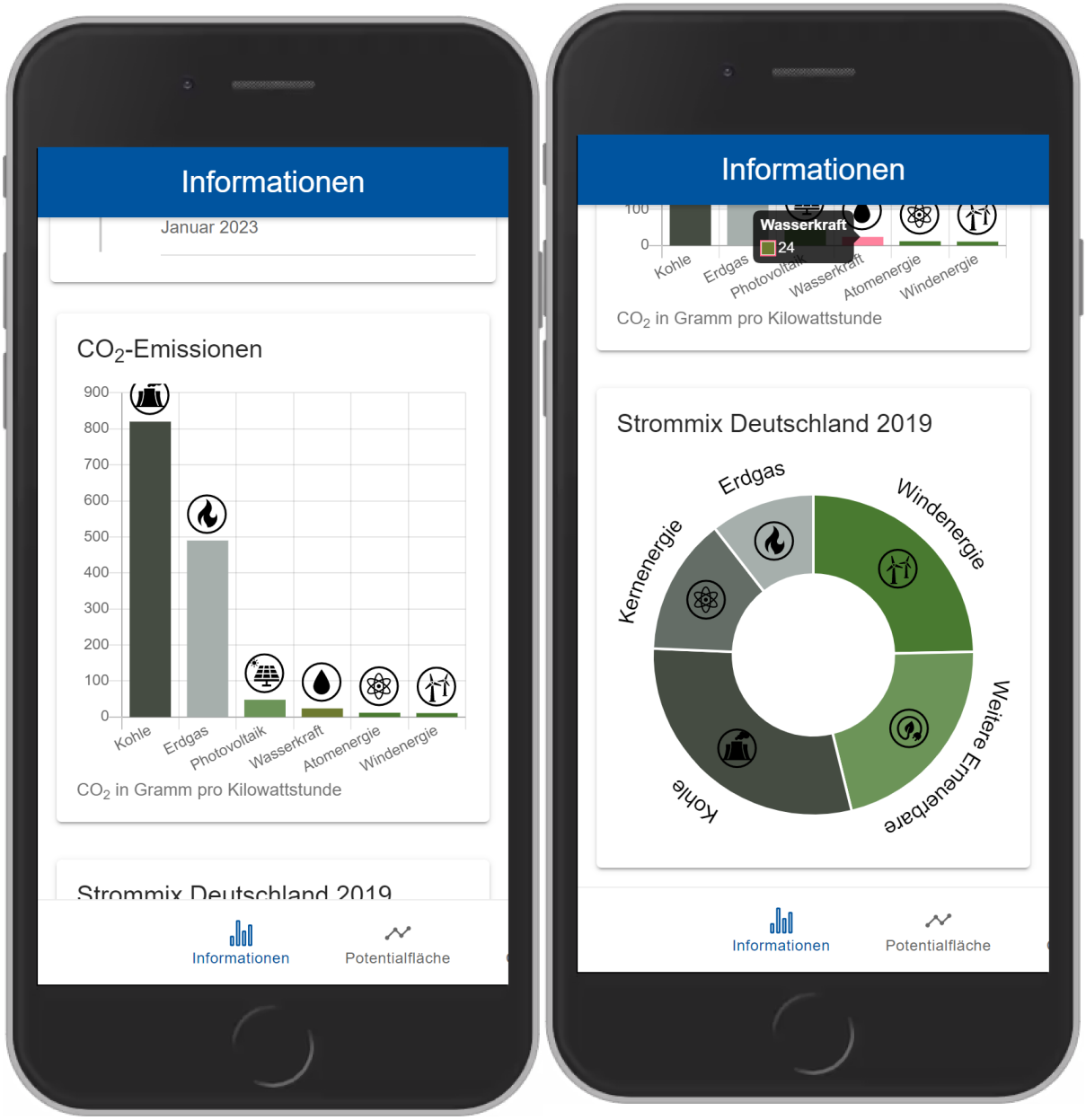
Although we have implemented asynchronous data delivery to improve the performance of the tool, the amount of imported information from all the sources and algorithms that we integrate could potentially be a cause for longer load times. We have attempted to cover some of the delays by implementing UI components, which clearly indicate when the user has interacted with them. In this way we are limiting the unpleasant user experiences, caused by confusion as a result of the longer waiting times.




(a) Card showing the different turbine types and their respective measures for the chosen scenario. This component also features an animated image of the turbine with its measures.

(b) Card showing the planned schedule for the construction of the wind farm and the different milestones which have to be achieved before the wind farm is put into production.

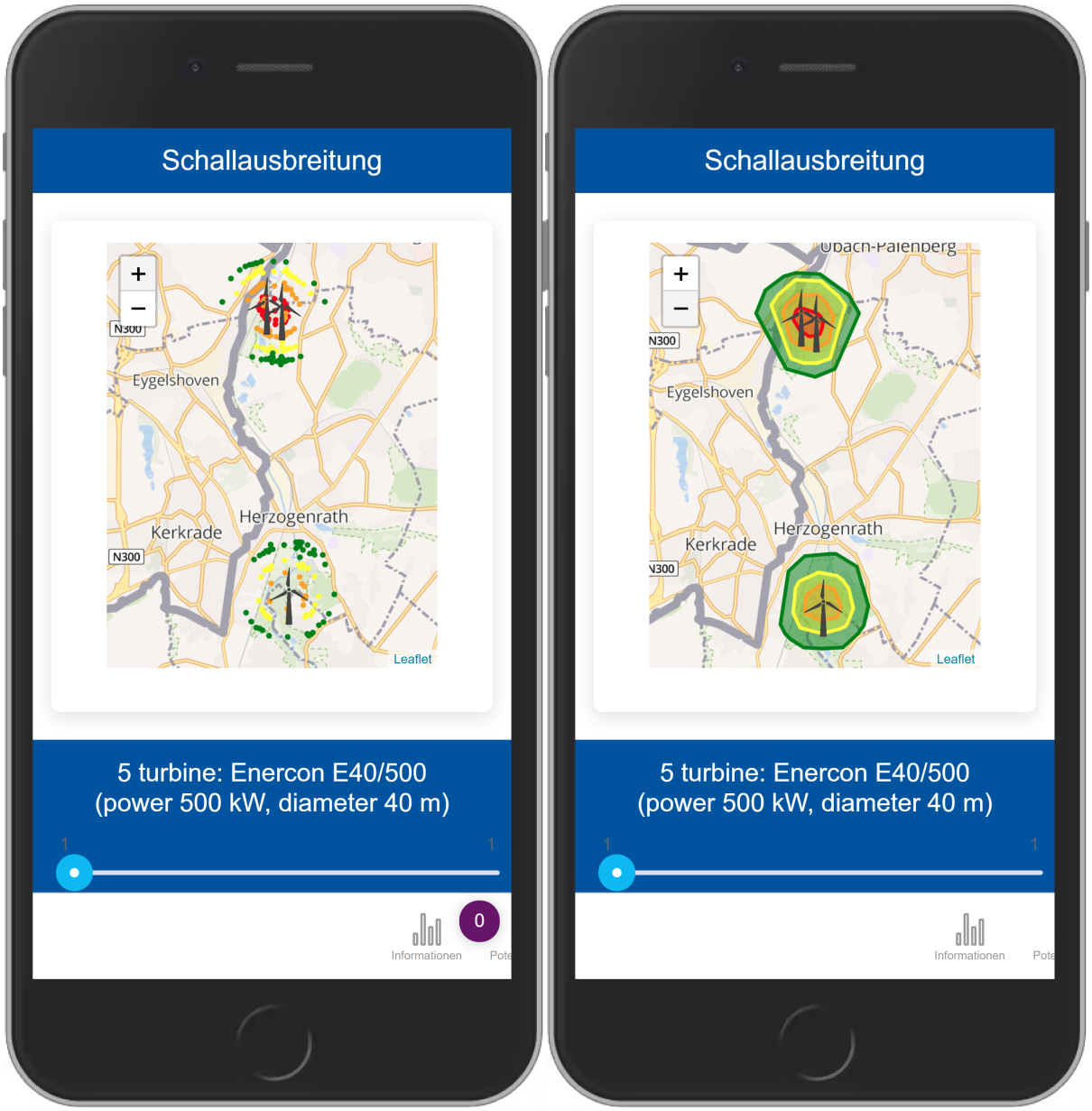
Figure 15: Outlook of a page, aimed at presenting general data about the wind farm, featuring turbine types, construction schedule. This is the first part of screenshots of this page, the next part is shown on Figure 16 (The scenario selection component is hidden for better visibility).




(a) Card showing a bar chart of the current $CO_2$ emissions by energy production source. Data is shown in grams per kilowatt-hour.

(b) A doughnut chart showing the distribution of electricity consumption according to the different sources of its production.

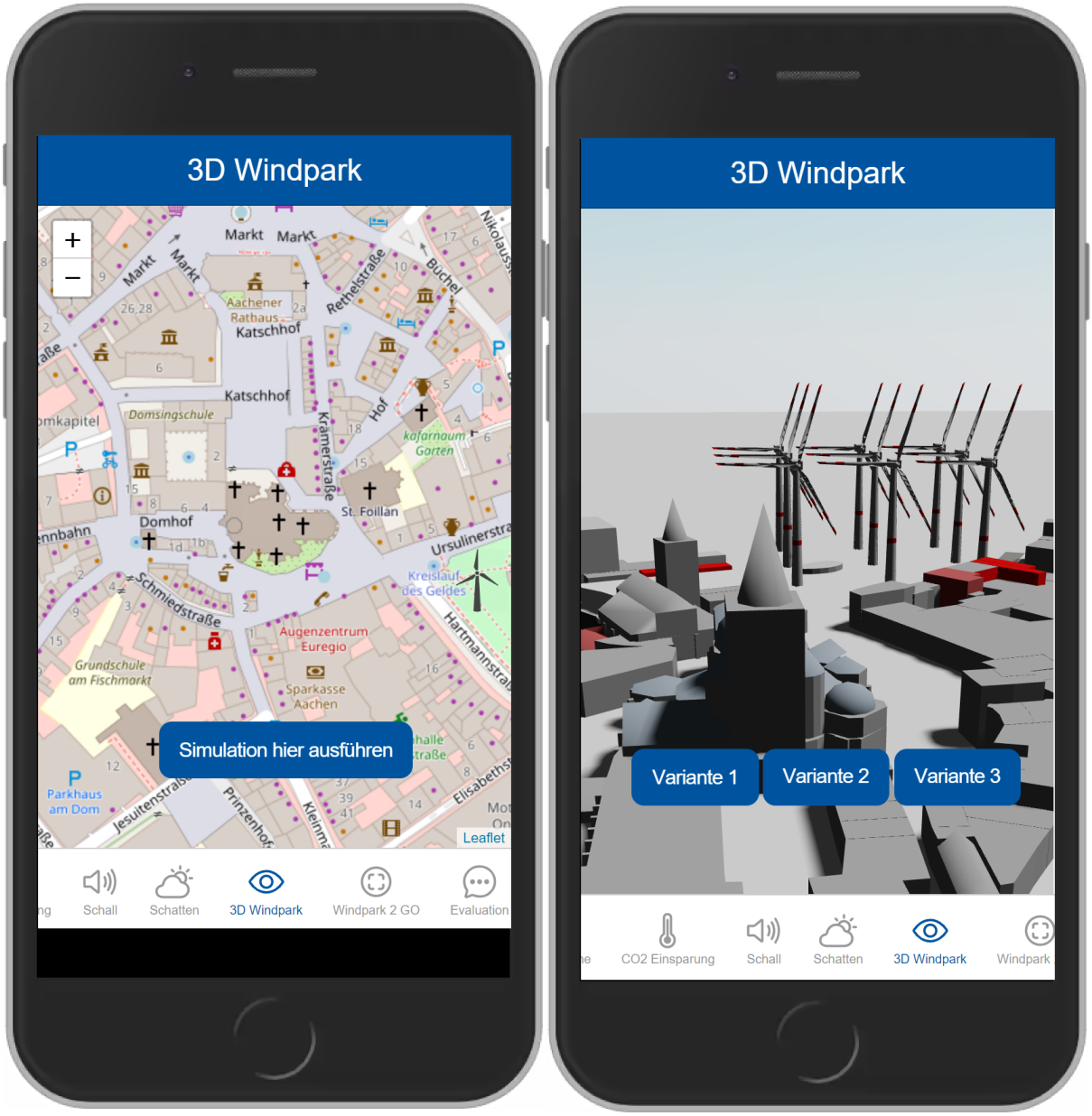
Figure 16: Outlook of a page, aimed at presenting general data about the wind farm, featuring a chart of the currently observed $CO_2$ emissions by energy production source and a doughnut chart of the distribution of electricity by the type of its source. The currently shown data used for the diagrams is not taken from a real source and is generated for displaying purposes. This is the second part of screenshots of this page, the first part is shown on Figure 15 (The scenario selection component is hidden for better visibility).




(a) Outlook of the page when unordered data is received.

(b) Outlook of the page when the border coordinates data is ordered into polygons.

Figure 17: A page meant for the presentation of our noise propagation algorithm results. This page features a map with the noise colored predictions we have generated for different locations. The map also shows the placement of the chosen wind farm scenario.




(a) An example of the map from which a location from the 3D scene generation has to be chosen. The map features an icon marker of the wind farm scenario, which is chosen and a button to confirm the choice of location and begin the simulation generation process.

(b) An example of an already generated 3D scene featuring a virtual wind farm.

Figure 18: Example for our 3D simulation page.

# 6 Conclusion

The goal of this work was to develop a tool which would be suitable for informing residents, living near an area, planned for the construction of a wind farm, on the different consequences that might be presented by such a project. In order to do this, we have implemented a platform where the users can get simplistic representations of the different technical data, produced by measurement and approximation, describing the effects a wind farm could have on the surrounding environment. We have made the tool easy to use and accessible on all popular devices. The approaches we have implemented present the different consequences of a wind farm in a manner, that would give the user a realistic perspective of the disturbance severity. The option for discovering the effects of different scenarios of the projects would also ease the distinguishing between the tradeoffs which have to be undertaken during the planning phase of such a project.

We have introduced both visual as well as aural simulation of the wind farm and its impact on the surrounding area. For these parts of our tool we had to develop special algorithms, which analyzed and computed, based on complex quantitative information, suitable data for its user friendly presentation. The algorithms use realistic data gathered from sources like OpenStreetMap and approximate the behavior of the aural and visual impacts of a wind farm. The results were stored and transferred to the frontend user interface of the tool and presented there.

Overall the accessibility and interactable nature of our tool would allow for a lowering in the informational gap between residents and wind farm owners. It is common for people who do not have easy access to such technical data to have difficulties distinguishing between aspects of the wind farm, that would cause them disturbances and such, that would not or would be negligible. Our tool would hopefully allow the local people to make an informed decision on the scenario of wind farm, they would find most suitable. As a consequence we are hypothesizing that the number of complaints and the amount of negativity against the wind farms would get lowered. This, however, remains to be quantified in the future.

## 6.1 Further Development

This work was just one part of a bigger project and as such could not cover all the challenges and wanted features under its scope. There are some possible next steps which could be implemented to better the performance of the separate working parts featured into our tool, or to introduce new behaviors to it.

1. Introduce parallelization or more complex shapes for the buildings into the noise propagation algorithm to better its performance or accuracy respectively.

2. Compute the final set of points into a concave instead of convex hull to improve precision of the final renders.

3. At the moment we are presenting the aural simulation only as visual components in our application. An addition of special audio effects, related to our noise propa-

gation estimation output data, into the visual 3D simulation could make the experience more well rounded and believable.

4. Introduce a structure that would allow for users feedback to be collected, processed and analyzed, as well as a way for this feedback to be presented to the companies and investors behind a specific wind farm construction project.

5. Optimize the load times of the frontend by using HTTP/2 where possible, lowering the number of big informational blocks, which have to be loaded at once. Some of the data can be stored into the local cache of the user in order to reduce load time significantly. High responsiveness and quick changes are very important for the feel and usability of mobile applications like ours.

6. Improve design decisions and UX even further, by bettering responsive behavior of UI components, render component placeholders when the data is still loading, introducing gesture events, which would better the navigation throughout the application.

# References


[1] Angular. URL https://angular.io/.

[2] Observations: oceanic climate change and sea level - NERC Open Research Archive. URL http://nora.nerc.ac.uk/id/eprint/15400/.

[3] Chart.js | Open source HTML5 Charts for your website. URL https://www.chartjs.org/.

[4] Comparing Cross-Platform Frameworks - Ionic - The Cross-Platform App Development Leader. URL https://ionicframework.com/resources/articles/ionic-vs-react-native-a-comparison-guide.

[5] Deno - A secure runtime for JavaScript and TypeScript. URL https://deno.land/.

[6] Electricity production from oil, gas and coal sources (% of total) | Data. URL https://data.worldbank.org/indicator/EG.ELC.FOSL.ZS?end=2015{&}start=1960{&}view=chart.

[7] Flutter - Beautiful native apps in record time. URL https://flutter.dev/.

[8] NgRx. URL https://ngrx.io/.

[9] What is Hybrid App Development? - Ionic - The Cross-Platform App Development Leader. URL https://ionicframework.com/resources/articles/what-is-hybrid-app-development.

[10] Bottom navigation - Material Design. URL https://material.io/components/bottom-navigation{#}usage.

[11] The most popular database for modern apps | MongoDB. URL https://www.mongodb.com/.

[12] Native mobile apps with Angular, Vue.js, TypeScript, JavaScript - NativeScript. URL https://nativescript.org/.

[13] Angular 2 Charts Demo. URL https://valor-software.com/ng2-charts/.

[14] NgRx. URL https://ngrx.io/.

[15] Node.js. URL https://nodejs.org/en/.

[16] About Objective-C. URL https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html.

[17] React – A JavaScript library for building user interfaces, . URL https://reactjs.org/.

[18] React Native · A framework for building native apps using React, . URL https://reactnative.dev/.

[19] What is REST API (RESTful API)? URL https://searchapparchitecture.techtarget.com/definition/RESTful-API.

[20] Bottom navigation - Material Design. URL https://material.io/components/bottom-navigation{#}usage.

[21] Swift - Apple Developer. URL https://developer.apple.com/swift/.

[22] UI Components - Ionic Documentation. URL https://ionicframework.com/docs/components/.

[23] Vue.js. URL https://vuejs.org/.

[24] Web project: Why use a Framework? URL https://www.bocasay.com/why-use-framework/.

[25] What is a wind class? URL https://www.lmwindpower.com/en/stories-and-press/stories/learn-about-wind/what-is-a-wind-class.

[26] Xamarin | Open-source mobile app platform for .NET. URL https://dotnet.microsoft.com/apps/xamarin.

[27] Xcode 12 - Apple Developer. URL https://developer.apple.com/xcode/.

[28] Wind Energy Modeling and Simulation | Wind | NREL. URL https://www.nrel.gov/wind/modeling-simulation.html.

[29] OpenStreetMap. URL https://www.openstreetmap.org/{#}map=6/51.330/10.453.

[30] overpass turbo. URL https://overpass-turbo.eu/.

[31] Technology – wind farm Monitoring. URL https://www.softvise.de/en/technology{_}wpm.html.

[32] Three.js – JavaScript 3D library. URL https://threejs.org/.

[33] Introducing the WebVR 1.0 API Proposal - Mozilla Hacks - the Web developer blog. URL https://hacks.mozilla.org/2016/03/introducing-the-webvr-1-0-api-proposal/.

[34] Schalltechnisches gutachten für die errichtung und den betrieb von vier windenergieanlagen am standort wessin. Technical report, Wind GmbH and Co. KG, 2018.

[35] Schalltechnisches gutachten für die errichtung und den betrieb von vier windenergieanlagen am standort güsten. Technical report, Wind GmbH and Co. KG, 2019.

[36] Roalt Aalmoes and Merlijn Boer, den. Simulation of wind turbine noise for community engagement. *The Journal of the Acoustical Society of America*, 141(5): 3805–3805, may 2017. ISSN 0001-4966. doi: 10.1121/1.4988406. URL `http://asa.scitation.org/doi/10.1121/1.4988406`.

[37] Luis Barrios and Alejandro Rodríguez. Behavioural and environmental correlates of soaring-bird mortality at on-shore wind turbines. *Journal of Applied Ecology*, 41 (1):72–81, feb 2004. ISSN 00218901. doi: 10.1111/j.1365-2664.2004.00876.x. URL `http://doi.wiley.com/10.1111/j.1365-2664.2004.00876.x`.

[38] Robert Berry, Gary Higgs, Richard Fry, and Mitch Langford. Web-based GIS Approaches to Enhance Public Participation in Wind Farm Planning. *Transactions in GIS*, 15(2):147–172, apr 2011. ISSN 13611682. doi: 10.1111/j.1467-9671.2011.01240.x. URL `http://doi.wiley.com/10.1111/j.1467-9671.2011.01240.x`.

[39] Franck Bertagnolio. A noise generation and propagation model for large wind farms. Technical report.

[40] Francesco Bosello, Roberto Roson, and Richard S.J. Tol. Economy-wide estimates of the implications of climate change: Sea level rise. *Environmental and Resource Economics*, 37(3):549–571, jul 2007. ISSN 09246460. doi: 10.1007/s10640-006-9048-5.

[41] Jens H. Christensen and Ole B. Christensen. Severe summertime flooding in Europe. *Nature*, 421(6925):805–806, feb 2003. ISSN 00280836. doi: 10.1038/421805a.

[42] Pietro Cipresso, Irene Alice Chicchi Giglioli, Mariano Alcañiz Raya, and Giuseppe Riva. The past, present, and future of virtual and augmented reality research: A network and cluster analysis of the literature. *Frontiers in Psychology*, 9(NOV), nov 2018. ISSN 16641078. doi: 10.3389/fpsyg.2018.02086.

[43] Gerald Dekker, Qiuhao Zhang, John Moreland, and Chenn Zhou. MARWind: Mobile Augmented Reality Wind Farm Visualization. Technical report.

[44] DIN IEC 60651. Sound level meters. Standard, DIN Deutsches Institut für Normung e. V., DIN German Institute for Standardization, 1980.

[45] DIN ISO 9613-2. Acoustics - attenuation of sound during propagation outdoors - part 2: General method of calculation. Standard, DIN Deutsches Institut für Normung e. V., DIN German Institute for Standardization, 1997.

[46] Con Doolan. A Review of Wind Turbine Noise Perception, Annoyance and Low Frequency Emission. *Wind Engineering*, 37(1):97–104, feb 2013. ISSN 0309-524X. doi: 10.1260/0309-524X.37.1.97. URL `http://journals.sagepub.com/doi/10.1260/0309-524X.37.1.97`.

[47] Caroline Draxl, Andrew Clifton, Bri Mathias Hodge, and Jim McCaa. The Wind Integration National Dataset (WIND) Toolkit. *Applied Energy*, 151:355–366, aug 2015. ISSN 03062619. doi: 10.1016/j.apenergy.2015.03.121.

[48] M. D. Flannigan, B. J. Stocks, and B. M. Wotton. Climate change and forest fires. *Science of the Total Environment*, 262(3):221–229, nov 2000. ISSN 00489697. doi: 10.1016/S0048-9697(00)00524-6.

[49] R. L. Graham. An efficient algorith for determining the convex hull of a finite planar set. *Information Processing Letters*, 1(4):132–133, 1972. ISSN 00200190. doi: 10.1016/0020-0190(72)90045-2. URL `http://www.math.ucsd.edu/{~}ronspubs/72{_}10{_}convex{_}hull.pdf`.

[50] Malcolm D Hayes and Hayes Mckenzie Partnership. The Measurement of Low Frequency Noise at Three UK Wind Farms. Technical report, 2007. URL `http://www.telegraph.co.uk/news/main.jhtml?xml`.

[51] im Studiengang Bachelor, Betreuende Prüferin, Ulrike Steeens Zweitgutachter, Martin Behrmann geb Knoblauch, and Eric Wohlgethan. Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung. Technical report.

[52] Slavina Ivanova and Georgi Georgiev. Using modern web frameworks when developing an education application: A practical approach. In *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2019 - Proceedings*, pages 1485–1491. Institute of Electrical and Electronics Engineers Inc., may 2019. ISBN 9789532330984. doi: 10.23919/MIPRO.2019.8756914.

[53] Şenay KOCAKOYUN. Developing of Android Mobile Application Using Java and Eclipse: An Application. *International Journal of Electronics, Mechanical and Mechatronics Engineering*, 7(1):1335–1354, jun 2017. ISSN 21460604. doi: 10.17932/iau.ijemme.21460604.2017.7/1.1335-1354.

[54] Ulf Liebe, Anna Bartczak, and Jürgen Meyerhoff. A turbine is not only a turbine: The role of social context and fairness characteristics for the local acceptance of wind power. *Energy Policy*, 107:300–308, aug 2017. ISSN 03014215. doi: 10.1016/j.enpol.2017.04.043.

[55] ANDREW LOTHIAN. Scenic Perceptions of the Visual Effects of Wind Farms on South Australian Landscapes. *Geographical Research*, 46(2):196–207, jun 2008. ISSN 1745-5863. doi: 10.1111/j.1745-5871.2008.00510.x. URL `http://doi.wiley.com/10.1111/j.1745-5871.2008.00510.x`.

[56] Nikolai Mina. GAME ENGINES FOR ARCHVIZ: THE FUTURE OF ARCHITECTURAL VISUALISATION? 2019.

[57] M. Monirul Qader Mirza. Climate change, flooding in South Asia and implications. *Regional Environmental Change*, 11(SUPPL. 1):95–107, dec 2011. ISSN 14363798. doi: 10.1007/s10113-010-0184-7.

[58] Naomi Oreskes. Beyond the Ivory Tower: The scientific consensus on climatic change, dec 2004. ISSN 00368075.

[59] Philip Paar and Adrian Herwig. Game Engines: Tools for Landscape Visualization and Planning? Technical report. URL http://www.zalf.de/dbu-vis/.

[60] Jitu Padhye and Henrik Frystyk Nielsen. A comparison of SPDY and HTTP performance, jul 2012. URL https://www.microsoft.com/en-us/research/publication/a-comparison-of-spdy-and-http-performance/.

[61] Jin-seo Park. A New Concave Hull Algorithm and Concaveness Measure for n-dimensional Datasets *. Technical report. URL http://user.dankook.ac.kr/.

[62] Eja Pedersen, Frits van den Berg, Roel Bakker, and Jelte Bouma. Response to noise from modern wind farms in The Netherlands. *The Journal of the Acoustical Society of America*, 126(2):634–643, aug 2009. ISSN 15208524. doi: 10.1121/1.3160293. URL http://asa.scitation.org/doi/10.1121/1.3160293.

[63] Joseph Ratcliffe and Alain Simons. How can 3d game engines create photo-realistic interactive architectural visualizations? pages 164–172, 10 2017. ISBN 978-3-319-65848-3. doi: 10.1007/978-3-319-65849-0_17.

[64] Fritz Reusswig, Florian Braun, Ines Heger, Thomas Ludewig, Eva Eichenauer, and Wiebke Lass. Against the wind: Local opposition to the German Energiewende. *Utilities Policy*, 41:214–227, aug 2016. ISSN 09571787. doi: 10.1016/j.jup.2016.02.006.

[65] Hannah Ritchie and Max Roser. Renewable energy. *Our World in Data*, 2020. https://ourworldindata.org/renewable-energy.

[66] Elar Saks. JavaScript Frameworks: Angular vs React vs Vue. Technical report, 2019. URL http://www.theseus.fi/handle/10024/261970.

[67] Philipp Schulz. 3D-Visualisierung von OpenStreetMap GIS-Daten in WebGL 3D Visualization of OpenStreetMap GIS Data in WebGL. Technical report, 2020.

[68] Marco Sonnberger and Michael Ruddat. Local and socio-political acceptance of wind farms in Germany. *Technology in Society*, 51:56–65, nov 2017. ISSN 0160791X. doi: 10.1016/j.techsoc.2017.07.005.

[69] TNI ISO/TR 17534-3. Acoustics - software for the calculation of sound outdoors - part 3: Recommendations for quality assured implementation of iso 9613-2 in software according to iso 17534-1. Standard, DIN Deutsches Institut für Normung e. V., DIN German Institute for Standardization, 2015.

[70] OAR US EPA. Sources of Greenhouse Gas Emissions.

[71] VDI 4104. Sound propagation outdoors in consideration of meteorological and topographical conditions – part 2: Wind turbines. Standard, DIN/VDI-Normenausschuss Akustik, Lärmminderung und Schwingungstechnik (NALS), 2020.

[72] Christian C. Voigt, Ana G. Popa-Lisseanu, Ivo Niermann, and Stephanie Kramer-Schadt. The catchment area of wind farms for European bats: A plea for international regulations. *Biological Conservation*, 153:80–86, sep 2012. ISSN 00063207. doi: 10.1016/j.biocon.2012.04.027.

[73] Björn Zehner. Interactive wind farm planning in a visualization center - giving control to the user. 01 2009.

[74] Matthew Huaiquan Zhang. Appendix II: IEC Classification of Wind Turbines. In *Wind Resource Assessment and Micro-Siting*, pages 269–270. John Wiley & Sons, Singapore Pte. Ltd, Singapore, jul 2015. doi: 10.1002/9781118900116.app2. URL `http://doi.wiley.com/10.1002/9781118900116.app2`.