
Bachelorarbeit

# Das Minimum Viable Dataset – Übersicht über Urbane Daten und ihrer Anwendungen.

Nico Müller

*Erstprüfer:*
Prof. Dr. Andreas Dengel,
AG Wissensbasierte Systeme

*Zweitprüfer:*
M. Sc. Falco Nogatz,
Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

28. Januar 2022


Bachelor Thesis

# The Minimum Viable Dataset – A Survey on Urban Data and Its Applications

Nico Müller

*Reviewer:*
Prof. Dr. Andreas Dengel,
AG Wissensbasierte Systeme

*Supervisor:*
M. Sc. Falco Nogatz,
German Research Center for Artificial Intelligence GmbH

January 28, 2022

**Abstract**

In the course of digitalization, there has been an increasing trend towards publishing open data by local authorities. This has enabled opportunities for urban applications that improve the quality of life for the citizens. However, many municipalities still suffer from missing ideas for possible applications, which might appear irritating as there are several community-driven implementations that are ready-to-use – given they are fed on the required data.

In this thesis, we address this problem by providing an overview of urban ready-to-go applications emerging from the *Code for Germany* (CFG) network. For these applications, we define a dataset that is required for their realization, which we call a smart city's *minimum viable dataset*. To serve as an example, we set up the three applications *Click That 'Hood*, *ParkenDD*, and *Mietmap* for the city of Kaiserslautern, identify missing data, and give an outlook on possibilities for future extensions.

# Eigenständigkeitserklärung

Ich versichere hiermit, dass ich die vorliegende Bachelorarbeit mit dem Thema „Das Minimum Viable Dataset – Übersicht über Urbane Daten und ihrer Anwendungen.“ selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen wurden, habe ich durch die Angabe der Quelle, auch der benutzten Sekundärliteratur, als Entlehnung kenntlich gemacht.

| | |
|---|---|
| (Ort, Datum) | (Unterschrift) |

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

**API** Application Programming Interface . . . . . 10

**BMI** Bundesministerium des Innern, für Bau und Heimat . . . . . 5

**CFG** Code for Germany . . . . . 12

**CSS** Cascading Style Sheets . . . . . 19

**CSV** Comma Separated Values . . . . . 23

**DOM** Document Object Model . . . . . 19

**GeoJSON** Geographic JSON . . . . . 18

**HTML** Hypertext Markup Language . . . . . 14

**JSON** JavaScript Object Notation . . . . . 16

**OGD** Open Government Data . . . . . 9

**OK Lab** Open Knowledge Lab . . . . . 12

**OKF** Open Knowledge Foundation . . . . . 11

**URL** Uniform Resource Locator . . . . . 35

**W3C** World Wide Web Consortium . . . . . 14

**XML** Extensible Markup Language . . . . . 14

# Chapter 1

# Introduction

This chapter provides an introduction to the topic and defines the intended goals of this thesis. In Section 1.1, we start with a motivation, which briefly outlines the background and the focus of this work. Subsequently, Section 1.2 presents the goals and addressed problems of this thesis. The chapter concludes with an overview of the following contents in Section 1.3.

## 1.1 Motivation

In recent years, Digitalization has become increasingly important. However, it is not only the economic and social sectors that are affected by the upheaval caused by the inclusion of digital technologies. The change is also opening up new opportunities in urban areas. The goal of the cities is to make effective use of digitalization in order to efficiently deploy the available resources [23]. This should finally end up in increasing the quality of life for the citizens and the protection of the environment.

Cities that pursue this principle are often referred to as *smart cities*. They attempt to create an intelligent network of urban infrastructures in order to make administrative processes more transparent. To achieve this networking, open data, interfaces, and open standards play a decisive role [1]. Supposing to gain insight into the administration, the resulting data must first be made publicly available. Moreover, when it comes to the exchange of data, interfaces are used to facilitate access.

During the last years, there has been an increasing global trend to publish municipal data by local authorities [38]. Many cities, but also countries, provide a huge amount of data on specific portals. Thereby, the data originates from a wide variety of areas, such as administration, healthcare, or mobility. Ideally, the data is available in a machine-readable format to enable easy integration into any application. However, many still suffer from missing ideas on how to use these data in a meaningful way. This is in particular irritating since there are numerous community-driven implementations that are ready-to-use – given they are fed on the required data.

In this work, we define a minimal viable urban set of data that usually already exists and collect ready-to-go applications. The *Minimal Viable Dataset* is a reference to the

concept of the *Minimal Viable Product* (MVP) developed by Frank Robinson in 2001 [45]. In contrast to the MVP, the scope of this thesis is more about data than a product. More precisely, we identify the minimal dataset needed to run the ready-to-go applications mentioned above. To this end, we first provide a survey on existing community-driven applications in the urban area. Furthermore, we use the example of Kaiserslautern to set up three of such applications by integrating the required data. Thereby, the selected projects represent different data or techniques, which are frequently used in applications based on open data. In this context, it is possible that the data is initially not provided by the city of Kaiserslautern. If this is the case, we develop a method to obtain the missing data. In the next section, we provide an overview of the goals and addressed problems of the thesis in more detail.

## 1.2 Goals and Addressed Problems

The main goal of this thesis is that another city, municipality, or similar can use the given information to get an overview of existing applications and possibilities. The respective city should get an understanding, which data, in which format should be included in the corresponding open data portal. Therefore, the three projects should serve as possible start projects to enter the topic, but can also be implemented to expand the existing portfolio of applications. In order to achieve this main goal, the individual objectives of the thesis are listed in the following:

**Depicting the state of the art**
: We give an overview of the basic concepts such as *smart city* and *open data*. Therefore, we will focus on the current state in Germany and in particular on the realization of the smart city concept in Kaiserslautern.

**Collect ready-to-go applications**
: As an increasing amount of urban data is published, the realization of smart projects that could improve urban life is becoming more important. However, there may be a lack of ideas for exciting projects. To address this problem, we give a survey of existing ready-to-go applications, which could be of interest, especially for the urban area.

**Illustration of typical methods and technologies**
: In this thesis, we will take a close look at the implementation process of three selected projects. These projects are based on different technologies, require different data or access specific interfaces for data retrieval. Within this thesis, we give an overview of typical methods that are frequently used in the context of community-driven projects.

**Define a minimal viable dataset**
: In the overview of ready-to-go applications, one of the priorities is the required data. Specifically for the three selected projects, we discuss the necessary data,

how to obtain it, and the appropriate format. We provide an impression of which data should be provided by the city in which way in order to develop meaningful projects.

In the next section, we give an overview of the following chapters.

## 1.3 Overview of the Thesis

In order to get familiar with some important terms that are used in this thesis, we introduce some related concepts in Chapter 2. These concepts include *smart city*, the realization of Kaiserslautern as a smart city, *open data*, and the relevance of the *community* in the smart city context. Next, in Chapter 3, we introduce the basic data formats *XML*, *JSON*, and *GeoJSON*, which are used to provide the data for the presented applications. Furthermore, the programming languages *JavaScript* and *Python* are introduced, which are applied for specific presented projects. In Chapter 4, we give an overview of existing ready-to-go applications, specifically addressing their state and the required data. Subsequently, we explain the deployment process of three selected projects in the following chapters. Thereby, these chapters all consist of a motivation, related work, the adaptation process, and an outlook on future work. Chapter 5 introduces the first application *Click That 'Hood*, which is based on geodata and tests the geographical knowledge of the users. Chapter 6 discusses the application *ParkenDD*, which displays the available parking spaces in the integrated city. Moreover, Chapter 7 presents the application *Mietmap*, which visualizes the rental prices of an integrated city on a heatmap based on real estate data. Finally, Chapter 8 summarizes the main outcomes of this thesis.

# Chapter 2

# Background

This chapter introduces some important terms and concepts which are relevant to this thesis. After defining the smart city concept in Section 2.1, we specifically address the city of Kaiserslautern in Section 2.2. In Section 2.3, we focus on the topic of open data forming the basis of smart cities. The chapter concludes with an introduction to the community in Section 2.4. Especially the *Open Knowledge Foundation*, which is intensively involved with the concepts of open data and smart cities, is analyzed.

## 2.1 Smart Cities

The topic of digitalization becomes increasingly important and is also having an impact on the areas of the city and urbanization. Here, we often come across the concept and goal of the smart city, which is being pursued by individual cities. Although this term is frequently used, there is still no clear and uniform definition of the concept, only a variety of different interpretations [15]. For example, Harrison et al. [28] give a possible definition that represents well the different areas to which a smart city is related:

> *"[C]onnecting the physical infrastructure, the IT infrastructure, the social infrastructure, and the business infrastructure to leverage the collective intelligence of the city"*

A smart city thus strives for intelligent networking of all areas of life and economy in the respective municipalities. This network is created through the exchange of data and information among each other. It aims to use existing resources as efficiently as possible and, by connecting them, to create new opportunities for improving the city [23]. In this regard, the use of new technologies, data platforms, and, especially, transparent administrative processes are crucial to coming closer to the concept of a smart city [51].

In the next subsections, we first consider one way to define a smart city based on certain characteristics in Section 2.1.1. In Section 2.2.2, we look at the funding of smart cities in Germany. Finally, in Section 2.1.3, we analyze how German cities fare in international smart city rankings.

### 2.1.1 Characteristics of a Smart City

The broad concept of the smart city leads to the problem of comparison. It is challenging to compare different cities. That is why Giffinger et al. [26] published a report in 2007 in which they initially defined six characteristics of a smart city for the comparison and evaluation of these. The defined characteristics are *Smart Economy*, *Smart People*, *Smart Governance*, *Smart Mobility*, *Smart Environment*, and *Smart Living*. In a more recent work by Meier and Portmann [37] in 2017, this idea was taken up and extended. In their paper, dimensions are referred to rather than characteristics, whereby the individual seven dimensions and their contents are presented in the following:

- **Smart Governance**
  - Political and legislative management
  - Innovative cooperation of the public administration and the private sector
- **Smart Citizen**
  - Participation of citizens in urban projects
  - Communities for the exchange of knowledge
- **Smart Education**
  - Access to digital learning resources
  - Location and time independent
- **Smart Living**
  - Digital health services
  - Innovative design of living spaces (smart lighting, . . . )
- **Smart Mobility**
  - Intelligent traffic management
  - Access to traffic information to increase sustainability
- **Smart environment**
  - Sustainable resource management (energy, garbage, transport, . . . )
  - Analysis of (sensor) data to create sustainable opportunities
- **Smart Economy**
  - Sharing of Resources
  - Sustainability-considering economic models

### 2.1.2 Funding in Germany

Smart city thus is a trend that aims to make the city "better", or more precisely, to make superior use of its opportunities and resources. For this reason, the *Bundesministerium des Innern, für Bau und Heimat* (BMI) is funding smart cities in Germany [12]. The BMI is a German federal authority. One of its tasks is to modernize the state and its administration. A central component of this is the *Smart City Dialog*, in which the opportunities and risks of digitalization are discussed at a municipal level [11]. For this purpose, the *Nationale Dialogplattform Smart Cities* was created in 2016. It is a platform consisting of nearly 70 national experts from cities, municipalities, associations, and other organizations addressing the topic of digitalization. Its homepage can be accessed at `https://www.smart-city-dialog.de/nationale-dialogplattform`. Based on the discussions on the platform, the *Smart City Charta* was created in 2017, which is primarily aimed at the federal and state governments [10]. With the goal of sustainable urban development, the charta contains four central guidelines:

- Digital transformation needs goals, strategies, and structures.
- Digital transformation needs transparency and participation.
- Digital transformation needs infrastructures, data and, services.
- Digital transformation needs resources, competencies, and Cooperation.

To be able to fulfill these guidelines, the BMI has been funding *Modellprojekte Smart Cities* since 2019 [12]. For the next ten years, the federal government will provide a budget of around 820.000.000 € to test the above concepts in numerous German cities, districts, and municipalities.

### 2.1.3 German Municipalities in the International Smart City Ranking

The funding helps German cities to expand their smart city infrastructure and thus compare themselves in the European and international areas. However, according to a study by *eco* and *Arthur D. Little* [29], the main cities mentioned in the global space are Berlin, Hamburg, and Munich. Based on that, they conclude that financial resources, population density, and a good fiber-optic network are crucial for the establishment of a smart city.

When considering various international smart city rankings, a German city is rarely in the top ten. On the *Smart City Governments* website, accessed at `https://www.smartcitygovt.com/202021-publication`, Singapore is ranked first in an international comparison. London (3rd place) and Barcelona (4th place) are the strongest European leaders here. Berlin in 23rd place is the first German city included in this ranking. One reason for this may be that German cities are moving only slowly from isolated pilot projects to the extensive expansion of platforms and projects [29]. However, the aforementioned BMI funding is intended to help realize the vision of German cities as leading smart cities in the near future.

After having defined the term "smart city", the next section takes a closer look at the city of Kaiserslautern in terms of establishing itself as a smart city.

## 2.2 The Realization of the Smart City Concept Based on Kaiserslautern

With a population of over 100,000 citizens in 2020 [50], Kaiserslautern can be categorized as a large city. This suggests that Kaiserslautern, like many other large cities, is striving for and has even realized the concept of the smart city. In fact, the quest for a smart or connected city is not new. Back in 2006, Kaiserslautern participated in the *T-City* competition organized by *Deutsche Telekom*, in which the city developed ideas for urban networking. In 2017, Kaiserslautern was one of five finalists in the *Digitale Stadt* competition organized by *Bitkom*.[1] The competition was held in 2016-2017 and evaluated the concepts of a digital city created by medium-sized cities. According to a publication by Bitkom [51], this achievement had a major impact on the digitalization work of the city of Kaiserslautern. The success of the competition was followed by the founding of *KL.digital GmbH* at the end of 2017. This subsidiary has set itself the task of implementing and accompanying digitalization in the city of Kaiserslautern as Bitkom states. According to Verlage et al. [53], in addition to KL.digital, the city has a *Chief Urban Officer* (CUO) and a *Chief Digital Officer* (CDO), who advise the mayor on an honorary basis. While the CDO further develops the city's digital mission statement, the CUO focuses predominantly on the changes in the cityscape resulting from the development of the digital city [51]. Another important instance is the digitalization department, which is responsible for internal coordination within the administration and for managing digitalization projects, as Verlage states.

In the following subsections, we first examine the city's mission statement in Section 2.2.1. Furthermore, Section 2.2.2 addresses the topic of funding. Section 2.2.3 concludes with an analysis of Kaiserslautern's model projects .

### 2.2.1 The Mission Statement in Relation to Smart City and Digitalization

In 2018, the mission statement "herzlich digitale Stadt Kaiserslautern"[2] was adopted as stated by a municipal publication [47]. The mission statement explicitly emphasizes that the citizen is at the center of the digitalization process so that the focus lies on improving the quality of life. However, there are more conditions specified in the publication of the mission statement. According to the publication, the upcoming digitalization projects of the city of Kaiserslautern should:

- not being driven for its own sake,
- create transparency and encourage citizens to participate in urban development,

[1] Bitkom e. V. is the digital association of Germany. See also `https://www.bitkom.org/Bitkom/Ueber-uns`.

[2] "cordial digital city Kaiserslautern" (translated by the author). "Herzlich" in this sense is to be understood as friendly and honest.

- be aimed at the *entire* population,
- be suitable for daily use,
- be transferable to other municipalities, and
- be accepted by the population.

The mission statement and the above-mentioned conditions for smart projects clarify that the social aspect of the city of Kaiserslautern is especially important.

### 2.2.2 Funding of the Smart City Concept in Kaiserslautern

Through the existing network of municipal companies, the *Technical University of Kaiserslautern*, and the *Hochschule Kaiserslautern*, the city has a solid foundation to meet the requirements of digitalization. In addition, there are two institutes like the *Fraunhofer Institute for Industrial Mathematics* (ITWM) and the *German Research Center for Artificial Intelligence* (DFKI), which can assist the city in its projects. Kaiserslautern also receives funding from the state of Rhineland-Palatinate for its development into a digital model city, as well as at the federal level from the *Bundesministerium für Bildung und Forschung* (BMBF) and the *Bundesministerium für Wirtschaft und Energie* (BMWi) [51]. Furthermore, according to KL.digital [33], Kaiserslautern has been a 5G model region since 2019 and is another member of the *Modellprojekte Smart Cities*. The website of herzlich-digital states that Kaiserslautern receives about 15 million Euros from the BMI for projects in the field of digitalization and smart cities.

### 2.2.3 Smart City Model Projects

Currently (as of 2021-12-01), Kaiserslautern has projects from ten different areas already implemented or planned, whereby a complete overview of the projects can be found at `https://www.herzlich-digital.de/ueber-uns/projekte/`. The areas include *Security*, *Health*, *IT Infrastructure*, *Education*, *Administration*, *Commerce*, *Data Platform*, *Energy & Environment*, *Traffic* and *Society*. In the following, we consider some of these projects. One project that has already been realized is called *Mobile Glasfaser Infrastruktur* (MOGLI).[3] In cooperation with *empera*,[4] KL.digital has created a communication infrastructure, security forces can use to communicate at events. MOGLI thus forms an emergency infrastructure in case the conventional mobile networks are overloaded by unexpected events. This project increases the safety of the population, as it ensures the possibility of alerting, allowing security forces to respond in case of emergency. While this project is assigned to the topic area of security, a project from the area of society is currently being planned. The project is called *Stadt.Raum.Wir.*[5] and deals with the

[3] "Mobile fiber optic infrastructure" (translated by the author); See also `https://www.herzlich-digital.de/ueber-uns/projekte/mogli/`.

[4] *empera* is a brand of *K-net Telekommunikation GmbH*. They offer telecommunications solutions for Kaiserslautern and the surrounding area.

[5] "City.Room.Us." (translated by the author). See also `https://www.herzlich-digital.de/ueber-uns/projekte/stadt-raum-wir/`.

so-called "Third Space". In addition to the workplace (first room) and home (second room), the third room should be a public meeting place for everyone. According to KL.digital [33], this space is intended to promote social contact and make society more active, regardless of their generation. Since the amount of space in Kaiserslautern is limited, the idea is to allow citizens to interact and vote on what and for whom a popular space in Kaiserslautern should be used.

The model projects are intended to help improve the lives of the population. In order to implement such projects, data is necessary. For this reason, Kaiserslautern makes a number of datasets from different areas available in its open data portal, which can be accessed at `https://opendata.kaiserslautern.de/dataset`. Currently (as of 2021-12-01), 37 records can be found there, whereby the division into the respective categories is shown in Table A.1. The table shows that most datasets are available for the *Population and Society* sector, which includes 23 datasets. The available datasets are, for example, the age structure of the city of Kaiserslautern or the stock statistics. But also geodata of municipalities of Kaiserslautern or parking lot occupancies in machine-readable format can be found in the open data portal. This should enable the community to work on its own projects, which could benefit the city.

## 2.3 Open Data as a Basis for Smart Cities

Open data is a term that appears along with the concept of the smart city. According to the *Open Knowledge Foundation*, data is open "if anyone is free to access, use, modify, and share it – subject, at most, to measures that preserve provenance and openness" [25]. In our context of smart cities, the focus lies on data of municipalities and public government, for which the term *Open Government data* (OGD) is used [42]. According to the German government's open data strategy, this subcategory of open data is freely accessible and machine-readable data [13]. In addition, the data is not permitted to be personal and contain any information relevant to security.

The definition above summarizes the core of open data, but in the end, it is of limited help to the administration when it comes to making their data available in a useful way. For this reason, in 2007, a group of experts established eight principles for the publication of OGD, which is depicted at `https://opengovdata.org/`. Based on this, starting in 2010, the *Sunlight Foundation* expanded this list to ten principles [54]. The Sunlight Foundation is a nonprofit organization, that stands up for an open government. In the following, the elaborated principles of the living document are presented:

- **Completeness**
  - Data should be published as completely as possible
  - Metadata should be included
- **Primacy**
  - Published data should be primary source data

- **Timeliness**
  - Data should be published as soon as possible
- **Accessibility**
  - Data should be accessible to all without barriers
  - Data should be available via an *Application Programming Interface* (API)
- **Machine readability**
  - Data must be published in machine-interpretable formats
  - Documentation on the usage should be attached
- **Non-discrimination**
  - Anyone can access the data at any time
- **Open Standards**
  - Data should be provided in open standards
- **Licensing**
  - Licenses regulate the further use of published data
  - Data should be made as legally open as possible
- **Permanence**
  - Data that is published should remain published
- **Usage Costs**
  - Data should be provided free of charge

Having clarified what open data means and what requirements it has to fulfill, the question arises, why open data, respectively OGD is useful. For this purpose, the BMI [13] has focused on three central areas. The areas are economic growth, environmental and civic initiatives, and more efficient governance, which are explained in the following. According to BMI, open data opens up new application possibilities and business models, which can make Germany more economically attractive through new employment opportunities. Start-ups and medium-sized companies, which can profit from the publication of data, are particularly highlighted here. On the other hand, open data can help stakeholders to pursue their ecological goals by publishing environmental data. Furthermore, through portals such as `https://www.europeana.eu/en`, where more than 50 million digitized contents on books, music and works of art can be found, open data is also driving the cultural sector. Finally, the BMI emphasizes

the efficiency gains of the government. Public governance reduces search effort, facilitates coordination, and reduces duplication. Moreover, the resulting transparency makes decision-making processes more comprehensible, which ultimately benefits both citizens and the administration itself.

In addition to the potential benefits, the legal basis is of particular importance. With the *§ 12 a E-Government Act (EGovG)*[6] of 2017, the German federal government decided on the obligation to publish government data. In addition, according to BMI, since 2018 German authorities have had to make their data available to everyone, unprocessed and machine-readable, on the national metadata portal *GovData* [13], which can be accessed at `https://www.govdata.de/`. GovData is the national data portal for Germany, which provides a central access point for data from all levels of government [8]. In addition to the federal government, numerous states such as Baden-Württemberg, Bavaria, Berlin, Rhineland-Palatinate, etc. have joined the data portal and make their data available. To get an impression of the large amount of data, Table A.1 shows the current data inventory. According to GovData [27], its goal is to make government data from the federal, state, and local governments accessible in one place and thus easier to use. This provides a good focal point for open data that helps in the implementation of open data applications.

## 2.4 The Community as a Driving Force for Open Data and Smart Cities

For the full exploitation of the existing possibilities through open data and especially for the principle of a smart city, an active community is required. The interaction of many people enthusiastic about technology and digitalization and the open data provided results in the opportunity of developing many innovative and useful applications. The idea of the community is that experts, as well as newcomers, can exchange information about interesting projects and perspectives in order to jointly maintain existing and develop new project ideas. In a community, individual members do not need to have the same background. On the contrary, it is common for people with different educational and work backgrounds to pursue a shared goal together. Especially in the smart city use case, people with IT backgrounds work with government agencies to improve the city.

In the following two subsections, we first explain the important role of the Open Knowledge Foundation (Section 2.4.1). In Section 2.4.2, we end the chapter by taking a closer look at an event that both creates and strengthens a community, and thus has a major impact on the development of open data.

### 2.4.1 Open Knowledge Foundation

The *Open Knowledge Foundation* (OKF, `https://okfn.org/`) forms an important instance. The organization, founded in England, has set itself the task of creating a world in which all non-personal data is publicly available to everyone and knowledge is shared.

[6] E-Government-Gesetz – EGovG, `https://www.gesetze-im-internet.de/egovg/BJNR274910013.html`, Bundesamt für Justiz.

In order to achieve this, they have made it their mission to teach skills to work with data, create tools and services, and guide projects. Through their experience in technology and digitalization, they enable the implementation of many open data projects that benefit the governments, universities, and the population. They also drive the community forward by supporting networks like the *Open Knowledge Network* with their experience. With groups in more than 40 countries, the Open Knowledge Network enables people to communicate around the world, creating an international network.

In Germany, the OKF is represented by the *Open Knowledge Foundation Deutschland* (OKFD, `https://okfn.de/`) branch. In partnership with *Code for America* (CFA, `https://www.codeforamerica.org/`) and with the support of Google, OKFD operates the *Code for Germany* (CFG) program [17]. CFG is a network of experts from development, design, politics, government, humanities and more, who desire to advance open data, citizen participation, and government transparency together. This network is divided urbanely into local groups, so-called *Open Knowledge Labs* (OK Labs). In these, they develop digital tools for the urban space on a voluntary basis. The developed projects are then presented at `https://codefor.de/projekte/archiv/`. On this platform, numerous completed and also active projects can be found, which are divided into the categories of environment, politics, society, and mobility. The applications to be found there include visualizations of the government that are beneficial to citizens or useful software solutions based on mobility data. According to CFG [18], the demonstrated program aims to accomplish the following four objectives with its services:

- Develop further tools for citizens
- Drive open data and publish more data
- More visibility for local projects
- The exchange at the international level

Currently, CFG includes a total of 31 OK Labs. A full visual overview of the OK Labs can be found at `https://www.codefor.de/mitmachen/`. The OK Labs include major cities such as Hamburg, Berlin, Munich, and Frankfurt. Kaiserslautern itself has not (yet) founded an OK Lab, but representatives such as Saarland, Karlsruhe, and Heidelberg exist in this region.

### 2.4.2 Hackathons

In addition to the Open Knowledge Foundation and the associated codefor movement, hackathons are also held in Germany. For this, there are several dates in the year hackathons[7] with various topics are held. According to Briscoe and Mulligan [7], a hackathon is a topic-based programming event in which mainly programmers work collaboratively

[7] A list of hackathons that were organized in 2021 can be found at `https://www.hackathon.com/country/germany/2021`.

on software projects over a short period of time. From what was originally a small meeting among friends, its success has finally turned it into a serious and professional event that shines through its emerging projects.

Kaiserslautern also relies on the opportunities offered by the hackathon. Every year, Fraunhofer IESE organizes the so-called *PFAFF HACK* in Kaiserslautern. Participants spend a weekend on the Pfaff grounds developing ideas for solutions that will make the city of Kaiserslautern more environmentally and climate-friendly. The homepage with more information and the possibility of registration can be accessed at `https://pfaffhack.iese.de/de/`.

# Chapter 3

# Technologies

This chapter introduces some technologies that are fundamental for different applications of this thesis. Sections 3.1 and 3.2 present the two formats XML and JSON, which are used for structuring and exchanging text data. Following this, Section 3.3 discusses GeoJSON representing geographic data using JSON. Furthermore, two well-known programming languages JavaScript and Python are briefly introduced in Sections 3.4 and 3.5, since the applications discussed in this thesis are primarily based on them.

## 3.1 XML

The *Extensible Markup Language* (XML) is a markup language that is both machine and human-readable. Originating from an older format called *Standard Generalized Markup Language* (SGML), the XML 1.0 standard was published in 1998 by the World Wide Web Consortium[8] (W3C, [6]) in order to ease the implementation an be compatible with the *Hypertext Markup Language* (HTML) and SGML. XML is a text-based format that is primarily used for the exchange of structured information such as documents, data, books and more.

An XML document is made up of various components. Listing 3.1 shows a simple XML document that exemplifies a student's information in a structured way. Based on this example, the components of an XML document are explained in the following:

- **Introduction**

  Each XML document starts with an introduction (l. 1). The introduction contains:

  - the declaration of XML as language,
  - an annotation of the XML version, and
  - an annotation used for character encoding.

[8] The W3C is an international consortium made up of numerous organizations, developing web standards in cooperation with the public. For further information see also `https://www.w3.org/`.

- **Tag**

  Tags begin with a `<` and end with `>`. The `<address>` component of Listing 3.1 corresponds to a tag, whereby a distinction can be made between three different types:

  - Start tag (`<address>`)
    Start tag for the beginning of an element.
  - End tag (`</address>`)
    End tag for the ending of an element.
  - Empty tag (`<application/>`)
    Empty tags are used when elements have no content.

- **Element**

  Elements are the most important components for structuring an XML document. An element begins with a start tag, e.g. `<student>` and ends with an end tag, in this case `</student>`. If the value of an element is empty, it can also consist of an empty tag only. The value of an element can be:

  - text, e.g. `49777` is the value of the element `<mat></mat>`
  - attributes, e.g. `<name title="M.Sc.">`
  - other elements, e.g. `address><street>...</street></address>`
  - a mix of the above

- **Attribute**

  An element can contain several attributes. The name-value pair exists within a:

  - start tag, e.g. `<name title="M.Sc.">`.
  - empty tag, e.g. `<name title="M.Sc."/>`.

The code sample of Listing 3.1 is a well-formed XML document. To define what "well-formed" means, the W3C has specified guidelines [6]. The central aspects are mentioned in the following:

- An XML document must have exactly one root element.
- An XML document must have a corresponding end tag for all of its start tags.
- In each element, two attributes do not have the same name.
- XML elements have to be properly nested (no overlapping of elements).

**Listing 3.1:** Example of an XML document describing a student.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<student>
    <name title="M.Sc.">Mark Bauer</name>
    <university>TU Kaiserslautern</university>
    <mat>49777</mat>
    <address>
        <street>Max-Planck St.</street>
        <number>7</number>
        <city>Kaiserslautern</city>
    </address>
    <subject>Computer Science</subject>
    <application/>
</student>
```

So far we have discussed what components make up an XML document and which conditions have to be satisfied for it to be well-formed. But different ways exist to present the same information. For this reason, the structure of a document may vary strongly. This is where the Document Type Definition (DTD) comes into play. A DTD defines the document structure with a list of legal elements and attributes. However, according to Frank Neven et al. [4], the XML Schema Definition (XSD) is the most widely used language for defining XML structures since the DTD offers relatively few options for validation. Just as with a DTD, an XSD can be used to define a set of rules to check the validity of an XML document. In contrast to a DTD, an XSD is itself also an XML document. Another advantage is that in XSD primitive data types like string, datetime, integer, but also complex types like list and union are provided. In addition to XSD, there are other so-called *Schema Definition Languages*, such as *Relax ND* [16], *Schematron* [30] and more, which will not be discussed here in order not to exceed the scope of this thesis.

## 3.2 JSON

The *JavaScript Object Notation* (JSON) is a lightweight and simple text-based format for data exchange. It was initially introduced by Douglas Crockford [19] in 2006 and is currently specified in RFC 8259 [5]. Like XML, JSON is human-readable and easy for computers to parse and use. However, according to a study by Paulson et al. [39], JSON shows significantly better performance. Due to the better performance, but especially the well JavaScript integration, JSON has gained popularity in the web application area, and has become the main format for API communication [43].

Like XML, a JSON file consists of several components. Listing 3.2 shows a JSON object, which represents a possible way to demonstrates the XML document from List-

**Listing 3.2:** Example of a JSON object describing a student.

```
{
    "student": {
    "name": {
        "title": "M.Sc.",
        "name": "Mark Bauer"
    },
    "university": "TU Kaiserslautern",
    "mat": 49777,
    "address": {
        "street": "Max-Planck St.",
        "number": "7",
        "city": "Kaiserslautern"
    },
    "subject": "Computer Science",
    "application": []}
}
```

ing 3.1. Based on this example, the components of a JSON object are explained in the following:

- **Object**

  An object is surrounded by curly braces { and }. In Listing 3.2, the entry {`"title": "M. Sc.", "name": "Mark Bauer"`} is an example of an object. An object consists of key-value pairs (notation: `"key": value`), although an object can also be empty. Furthermore, it is specified that the key must be unique in each object.

- **Value**

  The value of the key-value pairs mentioned above can be of the following types:

  - Object
  - Array
  - Null: This allows to assign a null value to the key, which is to be distinguished from the numerical value 0 or the empty string "".
  - Boolean: This is represented by the keywords *true* and *false*.
  - String: A string is notated by double quotes "...". See `"Computer Science"` from the key `"subject"` in Listing 3.2.
  - Number: The number consists of the digits 0-9 and can contain a sign if required. See the matriculation number `49777` of the key `mat` in Listing 3.2.

- **Array**

  For the declaration of an array, an arbitrary number of values are written in square brackets `[` and `]`, separated by a comma. The type can be one of the values mentioned above (object, array, null, boolean, string, and number), but the array can also be empty. See the empty array `[]` of the key `application` in Listing 3.2.

In the next section, we introduce a format, which is based on the JSON.

## 3.3 GeoJSON

*Geographic JSON* (GeoJSON) is a simple JSON-based format for the representation of geodata and its properties. The format was initially published in 2008 and further specified in 2016 by the *Internet Engineering Task Force*[9] in RFC 7946 [14].

For the representation of geographic data, GeoJSON supports the geometry primitives *Point*, *LineString*, and *Polygon* which are introduced in the following:

- **Point**

```
{"type": "Point", "coordinates": [5.0, 10.0]}
```

  A point is defined by a single position.

- **LineString**

```
{"type": "LineString",
 "coordinates":
    [[10.0, 10.0], [10.0, 20.0], [20.0, 10.0]]}
```

  A LineString is defined by an array of positions.

- **Polygon**

```
{"type": "Polygon",
 "coordinates":
    [[[10.0, 20.0], [30.0, 40.0], [50.0, 68.0],
      [77.0, 20.0], [10.0, 20.0]]]}
```

  A Polygon is defined by a linear ring. A linear ring is a LineString with four or more positions, where the first and the last position is identical.

In addition, GeoJSON supports multipart geometries, which are composed of the primitives above. These include *MultiPoint*, *MultiLineString*, *MultiPolygon*, and *GeometryCollection*. The multipart geometries extend the primitives with a collection of their

[9] The Internet Engineering Task Force is an organization, that tries to improve the functioning of the internet. See also `https://www.ietf.org/`.

**Listing 3.3:** Example of a GeoJSON object.

```
{
    "type": "FeatureCollection",
    "features": [
        {
            "type": "Feature",
            "geometry": {
                "type": "Point",
                "coordinates": [7.769, 49.445]
            },
            "properties": {
                "someProperty": "someValue"
            }
        }, ... ]
}
```

respective type, while the GeometryCollection can contain different geometry types. In other words, for example, Multipoint has several points instead of one.

The key-value pair with the `"type"` key thus defines the presented type, where the value has to match one of the presented geometry types. Furthermore, `"coordinates"` contains the points represented, whereby the two elements in the square brackets correspond to either longitude and latitude, or easting and northing [14]. Thereby the order is fixed as indicated above and the coordinates must be given in decimal numbers. Moreover, altitude or elevation can optionally be specified as a third element.

So far we have only considered the individual geometry objects. In Listing 3.3, a code snippet of a GeoJSON object is shown. We distinguish between a *Feature* object and a *FeatureCollection* object. A FeatureCollection object has the type *FeatureCollection* and contains an array of several feature objects, which can also be empty. On the other hand, the type of a Feature object is *Feature*. Its contents include *geometry* and *properties*. Geometry corresponds to the geometry objects introduced above and properties refers to any JSON object.

## 3.4 JavaScript

JavaScript is a scripting language created specifically for the web. The language was announced by Netscape in 1995 and has become the standard for client-side scripting [31]. JavaScript is specified in ECMA-262, with ECMAScript 2021 currently being the most recent version [22]. Even though the name may let suggest that there is a strong connection to the *Java* programming language, the two languages were developed independently and the concepts differ fundamentally [3].

While HTML[10] and CSS[11] both handle the content and layout of a web page, JavaScript is used to manage the behavior of a web page. This behavior is caused by the interaction with the *Document Object Model* (DOM) which is facilitated by JavaScript. The DOM is a programming interface published by the W3C for structuring HTML and XML documents. DOM thus forms the connection between JavaScript and the web document. By representing the document as a tree structure, each node can be individually accessed and manipulated. Typical use cases for this are e.g. the control of specific buttons or the dynamic change of certain views over time. Here, an interpreter in the browser enables JavaScript code to be executed there directly without requiring an Internet connection [9].

Nowadays, JavaScript is not only used on the client-side anymore. Through environments like *Node.js* (`https://nodejs.org/`), it is possible to do server-side development with JavaScript. As applications developed with JavaScript became larger and more complex, so-called frameworks came into use. These should support the developer with basic code patterns so that the complete application does not have to be programmed from scratch. This allows the developer to focus on the individual sections and save time. Popular examples of such frameworks are *AngularJS* (`https://angularjs.org/`) and *Vue.js* (`https://vuejs.org/`).

## 3.5 Python

Python is a powerful programming language created by Guido van Rossum in the early 1990s as a successor to the *ABC* language. The Python interpreter, as well as other additional resources, are freely available at `https://www.python.org/`. Currently, Python 3.10 is the latest version.

Python is very popular due to its easy-to-learn syntax and readability. Its simplicity, along with its dynamic typing and interpretive nature, makes Python a commonly used language for rapid application development and scripting [52]. Furthermore, Python also supports object-oriented programming and relies on the concept of indentation. This means that e.g. instead of curly braces, indentation is used to group specific statements into a code block. Moreover, Python is platform-independent and offers numerous packages that make programming more comfortable. For example, Python is favored in the machine learning area, since packages as for example *NumPy* (Numeric Python) supports the handling of matrices and *Matplotlib* enables the plotting of graphics. For the installation of these packages the Python package installer *pip* is used. The latest version is pip 21.3.1, which can be accessed at `https://pypi.org/project/pip/`. By using the command *pip install* followed by the desired package name, the package can be downloaded and installed. Thereby the downloaded packages consist of different modules,

[10] *Hypertext Markup Language* (HTML) is a text-based markup language developed by the W3C that determines the structural design of a web page. The standard can be found here `https://html.spec.whatwg.org/`.

[11] *Cascading Style Sheets* (CSS) is the language for styling an HTML document. For further information see `https://www.w3.org/Style/CSS/`.

which can be integrated into the project via the keyword *import* followed by the module name.

Nevertheless, there are also disadvantages. For one thing, Python code is often slower in execution since the code is interpreted at runtime, and Python also does not support functionalities such as multi-threading.

# Chapter 4

# An Overview of Urban Data Applications

In Section 2.1, we mentioned that applications based on open data are crucial for the advancement of the smart city concept. In this context, we presented the CFG network, which builds a key factor in the development and maintenance of such applications. Currently, the CFG portal is accessible at `https://codefor.de/projekte/` and consists of 256 projects from various areas. However, not all of these projects are designed for urban areas, or be applicable to other cities.

In this chapter, we give an overview of some CFG projects, that are designed for the urban area and have already been accepted in different cities. In Table 4.1, we present the projects in tabular form including a short description, the required data, and an estimation for the effort of the respective adaptation. In addition to the brief description of the project, we refer to an active instance and name other cities that have successfully redeployed the respective application. Regarding the required data, we discuss which data is minimally necessary to set these applications up and which may need to be provided by the city. Additionally, we specify the appropriate format for the data provision, and mention if a scraper is needed for the extraction. Furthermore, we provide the individual projects with a state that indicates whether these are still maintained. Thereby, the state is clarified by the indication of the recent commit frequency of the respective project.

Reviewing the collected projects, a trend towards the visualization of points of interest on an interactive map can be identified. In this scenario, interactive maps should provide a quick and visual overview of the desired locations, as well as the ability to interact with the map to obtain additional information as needed. Furthermore, *green* applications with the goal to improve the environment are often implemented by cities, as well as applications that make urban government more transparent to the citizen.

**Table 4.1:** Tabular overview of urban CFG projects.

| **Project Name** | **Project Description** | **Required Data** | **State** |
|---|---|---|---|
| *Baumkataster* | In 2019, the OK Lab Karlsruhe developed an application named *Baumkataster* that visualizes the municipal trees. Baumkataster contains the coordinates and species of all municipal trees and displays the respective position on a map. In addition, the number of trees per municipality is set in relation to the number of inhabitants and the area in a graphical visualization. Furthermore, additional information about the individual trees can be retrieved. The instance of Karlsruhe can be accessed at `https://codeforkarlsruhe.github.io/baumkataster/`. Hamburg and Cologne have already implemented similar projects that deal with the visualization of tree cadastres. | Locations of the trees in GeoJSON format; municipal districts in GeoJSON format; population data per district e.g. in *Comma Separated Values* (CSV) format; Mapper for individual data must be adjusted **Effort: medium** | Active<br><br>Five commits in the last year. |
| *Wo ist Markt?* | In 2015, the OK Lab Karlsruhe developed an application called *Wo ist Markt?*, which visualizes various urban weekly markets. The required data is provided by the city. Meanwhile, the weekly market information can be retrieved for many cities including Leipzig, Munich, Kiel, Kaiserslautern and many more. A portfolio of all available cities can be accessed in the active version at `https://www.wo-ist-markt.de/`. | Weekly market data including names, dates, and locations in GeoJSON format. **Effort: low** | Finished, but still maintained<br><br>38 commits in the last three months. |
| *Wo ist Testzentrum?* | *Wo ist Testzentrum?* is an extension of the previously presented application *Wo ist Markt?* developed by the OK Lab Leipzig. The application is available since 2021 and visualizes the locations of *COVID-19* test centers in different German cities. An active instance can be accessed at `https://wo-ist-testzentrum.de/`. Currently, Berlin, Dusseldorf, Kalletal, Lemgo and the federal state of Hessen have integrated their data. | Locations of the COVID-19 test centers (JSON, GeoJSON) **Effort: low** | Active<br><br>17 commits in the last three months. |

| Project Name | Project Description | Required Data | State |
|---|---|---|---|
| *Was steckt in meinem Leitungswasser?* | In 2014, the OK Lab Heilbronnen developed an application that visualizes the ingredients of the tap water offered in their city. In addition, the offered tap water can be compared to various brands of mineral water available in stores, whereby the data is provided by the municipal utilities of Heilbronn. Meanwhile, numerous cities such as Leipzig, Berlin, and Cologne have adapted the project for their city. The active instance of Heilbronn can be accessed at `http://opendatalab.de/projects/trinkwasser/`. | Water data in a suitable format (JSON, CSV); city and district boundaries in GeoJSON format. Data must be integrated into a few files. **Effort: low-medium** | Seeks collaborators for extensions.<br><br>No commits in the last six years. |
| *Mietmap* | In 2015, the OK Lab Karlsruhe developed an application named *Mietmap*, which visualizes the rental price distribution of Karlsruhe. Therefore, the apartment offers are scraped from the real estate platform *ImmoScout24*.<br>The Mietmap is presented in more detail in Chapter 7 together with a description on how to adapt it for the city of Kaiserslautern. | Rental price data and/or scraper for ImmoScout24. **Effort: medium** | Finished<br><br>No commits in the last three years. |
| *Straßennamen* | Inspired by the application *Streetdudes* developed by *UlmApi*, the OK Lab Karlsruhe created a similar application in 2015. It visualizes the streets of Karlsruhe named after people and their genders. Furthermore, the application displays background information on the street names. The street names are scraped from a file provided by the city and the locations of the data come from *OpenStreetMap*. Both information are finally merged into a GeoJSON file. The instance of Karlsruhe can be reached at `https://codeforkarlsruhe.github.io/streetnames/`. Since 2020, the application has also been implemented for the city of Magdeburg. | The street names must be provided in an appropriate format (CSV, JSON, GeoJSON, etc.). Depending on the data format, the scraper must be adapted. **Effort: low-medium** | Finished, but still maintained<br><br>One commit in the last year. |

| Project Name | Project Description | Required Data | State |
|---|---|---|---|
| *Gieß den Kiez* | In 2019, the *Technologiestiftung Berlin* and the *CityLab Berlin* launched the project *Gieß den Kiez*, which can be translated into *water the district*. It is a platform where people can gather information about trees and their water resources in the city of Berlin. In addition, users can register and adopt individual trees, thus the users convey that they will water these trees regularly in the future. The active instance of Berlin can be found at `https://giessdenkiez.de/`. Meanwhile, the OK Lab Leipzig has adopted the project and is running an active version for their city. | Tree cadastre data in a suitable format (e.g. CSV, JSON, GeoJSON). The project consists of different repositories including backend, frontend, etc., which have to be adjusted. **Effort: high** | Active<br><br>16 commits in the last three months. |
| *Stolperstein-Bonn* | The OK Lab Bonn developed an application in 2015 that visualizes the urban stumbling blocks commemorating the victims of the National Socialism. The individual stumbling blocks are extracted from the Open Street Map database. Furthermore, they are linked to the respective Wikipedia articles to display additional information. The visualization of Bonn can be reached at `https://stolpersteine-in-bonn.de/`. Berlin and Bochum have already implemented this idea as a mobile application. The project information can be found at `https://codefor.de/projekte/stolpersteine-app/`. | City districts and boundaries in GeoJSON format. Further adjustments for the integration of the data and the extraction of the stumbling blocks have to be made. **Effort low-medium** | Finished<br><br>No commit in the last year. |
| *Klimawatch* | In 2019, the OK Lab Münster developed an application that compares the climate targets of the Paris Agreement with the actual emissions data of the municipality. For the visualization, the number of inhabitants and the $CO_2$ emissions of the municipality are required. Currently, in addition to Münster, numerous other German cities such as Karlsruhe, Cologne, Leipzig, etc. are represented in the application at `https://klimawatch.de/`. | Population and $CO_2$ emissions including year and category in CSV format. **Effort: low** | Active<br><br>Six commits in the last three months. |

| Project Name | Project Description | Required Data | State |
|---|---|---|---|
| *Altglas Container* | The OK Lab Berlin developed an application named *Altglas Container* in 2014, which visualizes the locations of waste glass containers of the Berlin district Charlottenburg-Wilmersdorf. An active instance can be accessed at `http://k-nut.github.io/altglas/`. | GeoJSON of the city boundaries and waste containers including the container category and manufacturer. **Effort: low** | Inactive<br><br>No commits in the last five years. |
| *Click That 'Hood* | *Click That 'Hood* is a game founded by the *Code for America 2013 Fellowship Team* in 2013, where people can playfully test their knowledge on the city's neighborhoods. Currently, numerous German cities such as Berlin, Hamburg, and Frankfurt have already been integrated Table B.1 shows all included German cities (and others), and a detailed project description is presented in Chapter 5. An active instance of Click That 'Hood can be accessed at `https://click-that-hood.com/`. | Map of the respective city including its subdivision into e.g. districts in GeoJSON format. **Effort: low** | Active<br><br>27 commits in the last three months. |
| *ParkenDD* | In 2015, the OK Lab Dresden developed the (mobile) application *ParkenDD*, which displays the current parking situation based on real-time data. Currently, the app includes twelve cities such as Dresden, Karlsruhe, Kaiserslautern, etc. on a stable basis. The project web page can be accessed at `https://parkendd.de/`. For further information we refer to Chapter 6. | Parking lot live-API (XML, HTML); parking lot locations (GeoJSON); scraper for data extraction. **Effort: low-medium** | Active<br><br>Four commits in the last three months. |

| Project Name | Project Description | Required Data | State |
|---|---|---|---|
| *Mat-O-Wahl* | *Mat-O-Wahl* is an internet-based program developed by Mathias Steudtner, which is intended to support the election decision. By means of theses, the own opinion can be compared to the party position. It is an alternative to the original *Wahl-O-Mat*, which is protected by copyright. The application is freely adjustable and suitable for mayoral elections, city council elections, district council elections and more. The original Mat-O-Wahl can be accessed at `https://mat-o-wahl.de/`. An adapted version has already been used for municipal elections in Wuppertal (2014), Bonn (2015) and Magdeburg (2019). | Questions and answers in CSV format; some changes in `data/definition.js`. Detailed step by step instructions available. **Effort: low** | Active<br><br>18 commits in the last three months. |
| *München Transparent* | *München Transparent* is a project of the OK Lab Munich, which was developed in 2013 and is running actively at the moment. The project is a council information system, which allows simplified access to information from the city council. The required data is offered via an *OParl* API. Currently, the OK Lab Munich is working on a flexible variant called *Meine Stadt Transparent*, which can be adapted to different cities. The instance of *München Transparent* can be found at `https://www.muenchen-transparent.de/`. | The council data must be provided via an OParl API. In addition, among other things, server and database changes have to be made, whereby a step-by-step guide can be found in the project description. **Effort: high** | Active<br><br>Three commits in the last year. |

The projects show that certain technologies and requirements occur frequently. These include the specification of locations using GeoJSON files and the provision of information via an API. Furthermore, it often comes to the usage of scrapers, which extract the required information from the provided file. In order to show the adaptation of such applications for new cities in more detail, we present the integration of Kaiserslautern into three of these applications in the following chapters. In Chapter 5, the integration of Kaiserslautern is shown by the example of *Click That 'Hood*. Click That 'Hood is a game based on geodata in GeoJSON format. In Chapter 6, the application *ParkenDD* is presented, which shows the available parking spaces with the help of a live-API in XML format. Finally, Chapter 7 introduces the interactive web application *Mietmap*, which uses a heatmap to display the urban rent distribution. The required data is obtained via a scraper, which extracts the necessary data from the real estate portal *ImmoScout24*. Chapters 5 to 7 are divided into a motivation, followed by related work and the final integration. At the end of each chapter, an outlook on future work is given.

# Chapter 5

# Click That 'Hood

In Chapter 4, we presented an overview of open data applications, which are particularly interesting in the context of smart cities. In this chapter, we take a closer look at a specific application named *Click That 'Hood*. Click That 'Hood is a game based on open data, which is available in the GeoJSON format. In Section 5.1, we give a motivation, in which the game and its relevance are introduced. Section 5.2 continues with related work focusing on GeoJSON in a municipal context. The integration of a new city into Click That 'Hood is presented in Section 5.3, using Kaiserslautern as an example. The chapter concludes with an outlook on possible future projects in Section 5.4.

## 5.1 Motivation

Click That 'Hood is a web application based on open data where people can playfully demonstrate their geographical knowledge. The open-source game was founded in 2013 by the *Code for America 2013 Fellowship Team*. The project started with the intention to better get to know the neighborhoods of the city of Louisville and the main developer's interest in geovisualization techniques [36]. Already after two years, the initially small project has developed into a large application consisting of more than 200 locations. Currently, over 270 geographic locations can be found. Furthermore, the categories have been expanded to include countries, states, continents, airports, and much more. However, the project has not been maintained or expanded in recent years. For this reason, prompted by this thesis and due to the ongoing and active interest in the community, Click That 'Hood is actively operated by *Code for Germany* (cf. Section 2.4.1) since September 2021. A public instance of Click That 'Hood can be found at `https://click-that-hood.com/`.

The game is based on geographical maps, which represent cities, countries, continents, or similar. Depending on the selected map, it is divided into certain areas. For example, countries can be separated by their states or federal states, while a city may be divided into its districts. In the end, the type of subdivision is up to the city's integrator. When a category is picked, the corresponding map with the selected subdivision is displayed. The goal of the game is now to identify and pick all the mentioned areas (cities,

countries, districts, ...) as fast as possible. In Figure 5.1, the start page of the game is shown. The page is visible after a city – in this case the city of Kaiserslautern – has been selected for a match. In the background, we can recognize the map of Kaiserslautern containing the outlines of different districts. The game is started by clicking the *Start* button.

The philosophy of the project is that the portfolio of offered maps and categories should be extended by the community. Click That 'Hood is based on the idea that interested people integrate e.g. their city into the project if it is not available yet. By now, a large number of cities, countries, etc. can be discovered. Besides new maps, another game mode has been added. Instead of only using shapes, specific locations can be highlighted using points of interest. In this way, for example, the 100 highest peaks in the world can be selected on a map. In Table B.1, we give an excerpt of playable maps including the number of requested areas.

Because of this playful character and the few resources it requires for integration, Click That 'Hood is a popular beginner project for interested developers and for cities venturing into the topic of open data and possible applications. Taking only the Germany as an example, numerous cities such as Berlin, Hamburg, and Frankfurt have already been integrated. For this reason, we select this project as the first example and explain how the integration of the city of Kaiserslautern based on its district boundaries works in Section 5.3.

## 5.2 Related Work

In this section, we provide an overview of topics related to Click That 'Hood. In Section 5.2.1, we present further municipal applications that are also based on GeoJSON. In Section 5.2.2, we introduce the principle of gamification, which is applied by Click That 'Hood.

### 5.2.1 Further Municipal Applications Using GeoJSON

In the following, we present two other applications that are also based on geodata in the GeoJSON format, possibly provided as open data. Firstly, we introduce *You Don't Know Africa*, which is a modified version of Click That 'Hood. And secondly, we present *Urban Viewer*, an application that visualizes the population of different cities.

#### You Don't Know Africa

You Don't Know Africa is a variation of Click That 'Hood developed by David Bauer. The game was created shortly after Click That 'Hood in 2013. An instance of the game can be accessed at `https://www.youdontknowafrica.com/2/`. Instead of a large selection of playable locations, the game contains exclusively the countries of Africa. The game principle remains the same. Based on a map of Africa, the requested countries have to be clicked as fast as possible. Bauer offers two modes of difficulty. In the simple case, the map is limited to 20 random countries, while in the difficult game, the entire

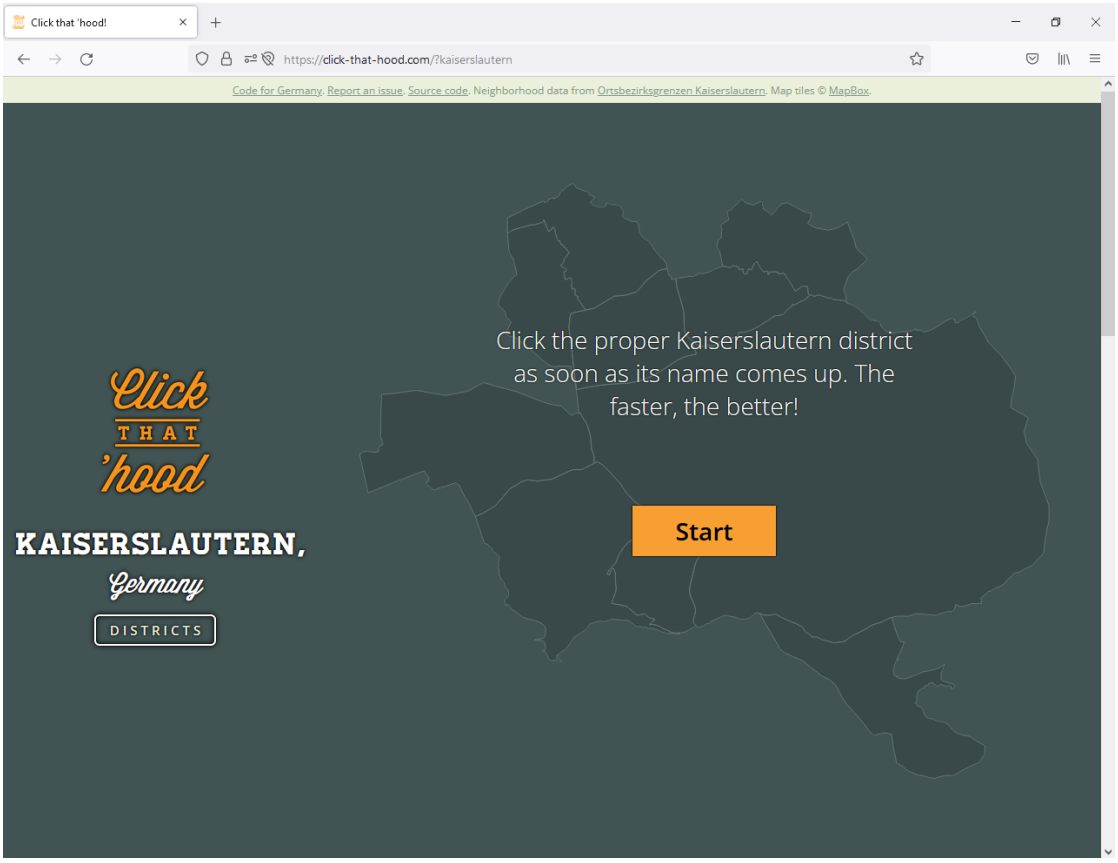



**Figure 5.1:** Start page of Click That 'Hood when Kaiserslautern is selected for playing.

map with all 51 countries is queried. Click That 'Hood has opted for tooltips, which display the names when hovering over the corresponding location. You Don't Know Africa explicitly foregoes these to make the game more challenging.

You Don't Know Africa received strong interest, and ended up being more popular than the original application [36]. Bauer[12] explains the success on the one hand by the good template Click That 'Hood and the challenging thesis of the title. On the other hand, the game choice is limited to Africa only. This limited choice allows players to better compare their experiences and results with others.

Since then, Bauer has developed a sequel to his first version in 2020, titled *You Still Don't Know Africa*, which has a slightly different game principle. Instead of clicking the correct countries on a map, 20 countries have to be written from memory within two minutes. For naming all countries, the player has ten minutes of time.

[12]Blog post "#YOUDONTKNOWAFRICA: Learnings from an accidental worldwide internet hit" (2013), `http://www.davidbauer.ch/2013/12/01/youdontknowafrica-learnings-from-an-accidental-worldwide-internet-hit/`, David Bauer.

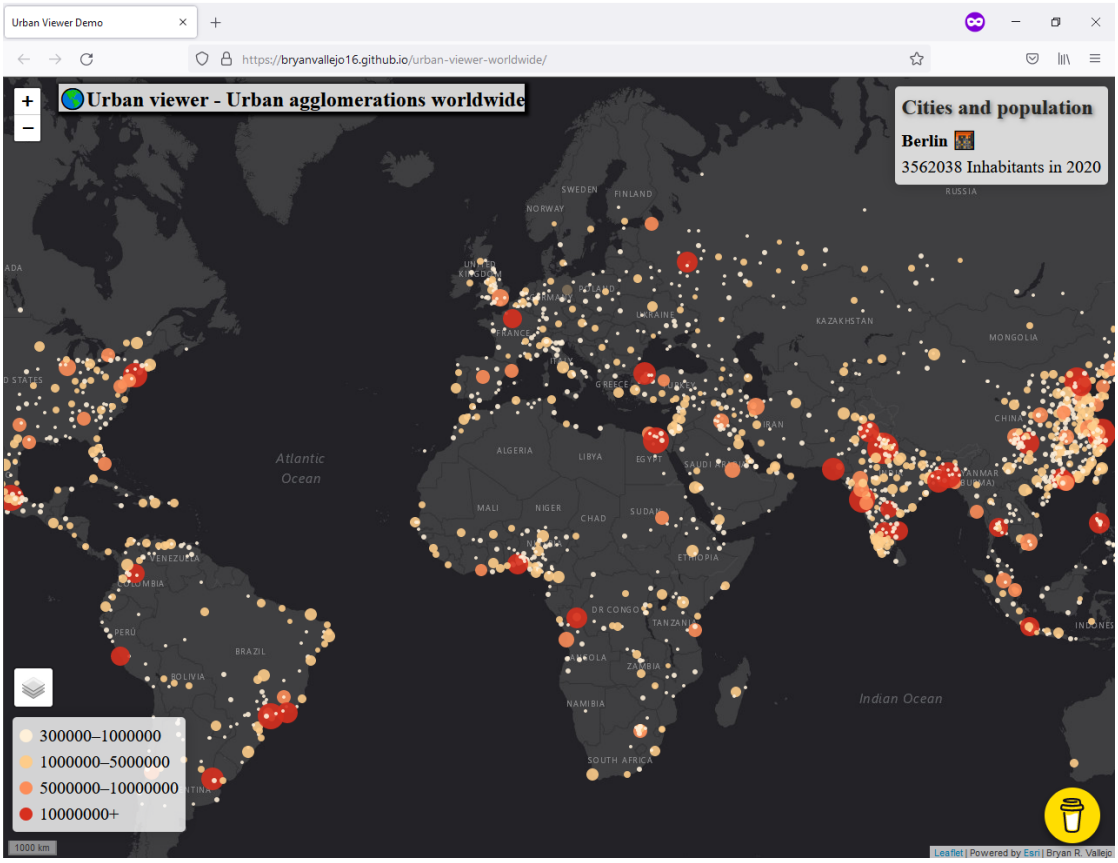



**Figure 5.2:** Screenshot of the Urban Viewer application.

### Urban Viewer - Urban Agglomerations Worldwide

Urban Viewer is an interactive web map that visualizes the population of urban agglomerations worldwide. The application by Bryan Vallejo[13] thus provides a good overview of the population in urban areas in 2020. An instance of the interactive web map can be accessed at `https://bryanvallejo16.github.io/urban-viewer-worldwide/`. Figure 5.2 shows a screenshot of the map. Placed on this map, there are points that vary in color and radius. The points represent the individual cities and visualize the extent of the population in the respective city by their size and color. By hovering over these points, the selected city and its population are displayed in the upper right field of the screen. In the case of Figure 5.2, Berlin is hovered. Vallejo obtains the required population data from the *Department of Economic and Social Affairs* of the United Nations, which can be accessed at `https://population.un.org/wup/Download/`. The points

[13]Blog post "Discover Urban viewer: an interactive web map to visualize population of Urban Areas Worldwide" (2022), `https://towardsdatascience.com/discover-urban-viewer-an-interactive-web-map-to-visualize-population-of-urban-areas-worldwide-f36df1fe420d`, Bryan Vallejo.

of each city are based on the GeoJSON format. Each city is represented by the point geometry presented in section Section 3.3.

### 5.2.2 Gamification in the Smart City Context

As we have seen in Kaiserslautern's realization of the Smart City concept (cf. Section 2.2), the citizen comes to the fore. In order to encourage the citizens' participation, the *gamification* approach comes into play. According to Deterding et al. [20], gamification can be defined as the "use of game design elements in non-game contexts". The term originally comes from the marketing and education sector. In recent years, it has gained significance especially in the urban context [40]. For instance, specific game mechanisms such as rankings, scores, levels, and badges could motivate people to actively participate in the smart city process. Since gamification has become more and more important for smart cities in recent years, various frameworks have been developed. These frameworks should simplify the integration of gamification into applications. One example is the framework of Kazhamiakin et al. [32], which can be applied to existing services and applications in the smart city context.

Currently, there are many applications in various domains that try to integrate the gamification concept. For instance, the city of Surrey has developed a game that playfully refreshes the rules of waste separation. In this game, which is called *Rethink Waste*, different types of waste are shown in sequence and have to be placed in the appropriate trash can. The hope is to increase awareness of the waste separation system, resulting in improved environmental friendliness. The game can be played at `https://surrey.recycle.game/`.

In this context, Click That 'Hood is a good example of a gamification application. Users can further improve their geographical knowledge in a playful way, for many people, Click That 'Hood is the first step into the smart city world. The game motivates people to integrate their own city and thus deal with open data applications and their underlying techniques, such as the GeoJSON data format. In this way, the playful character can generate experience and evoke interest in people, which may encourage them to participate in further projects.

## 5.3 Integration of Kaiserslautern

In the following, we take a look at the integration of new cities into Click That 'Hood, on the example of Kaiserslautern. It is worth noting that Kaiserslautern is just one example, that can be projected to any other city if the necessary data is provided. Furthermore, the technical realization of the application is only discussed superficially here because it is not the focus of this thesis. In Section 5.3.1, we first consider setting up the local environment of Click That 'Hood, which is required to integrate new cities. After that, in Section 5.3.2, we focus on the integration of the necessary data.

### 5.3.1 Setting up the Local Environment

Click That 'Hood has a GitHub repository that is freely accessible for everyone at `https://github.com/codeforgermany/click_that_hood`. The first step is to clone this project and set it up using the following commands:

```
1 $ git clone https://github.com/codeforgermany/click_that_hood.git
2 $ cd click_that_hood
3 $ npm install
4 $ npm run build
5 $ npm start
```

As expected from a web application, when we look at the project structure, the project mainly consists of JavaScript code, while the web interface is realized by HTML and CSS. A brief introduction to the concepts of JavaScript and the relationship to HTML and CSS can be found in Section 3.4.

The next step is to install the packages needed for the application. For this purpose, in line 2, we navigate into the cloned project. Next, in line 3 we install the required packages using the Node.js package manager *npm*. The package manager installs the dependencies defined in the *package.json* file. Node.js is introduced in Section 3.4. Moreover, in line 4, we load the already existing data, which consists of the above-mentioned cities, countries, and similar in GeoJSON format. The following problems may occur during the build process:

- *_Template.metadata.json* is located in the directory *data*.

  As the name indicates, this file provides a template on how the metadata for new cities should look structurally. However, during the build process, all files in *data* are crawled. To each *metadata.json* belongs a GeoJSON file, where the naming must be identical. In the case of the *_Template.metadata* file, no matching GeoJSON exists. To solve this problem the *_Template.metadata* file should be removed from the data directory.

- The GeoJSON file could not be found for some cities, countries, or similar.

  The error occurs because *data* initially contains a directory named *_work-in-progress*. The files in this directory are not correctly used for the build process. For this reason, the directory must also be removed from *data*.

When the build process has been completed error-free, the application can be started by `npm start` in line 5. By default, this makes the application locally accessible at `http://localhost:8020/`.

### 5.3.2 Integration of the Required Files

The relevant area for the integration of a new city is the *data* directory. Here, the data of all available maps can be found. For adding a new city, two related files are required. On the one hand, this is a JSON file containing the necessary metadata. The name

**Listing 5.1:** Metadata file of Kaiserslautern in Click That 'Hood.

```
{
    "locationName": "Kaiserslautern",
    "countryName": "Germany",
    "annotation": "Districts",
    "dataUrl": "https://opendata.kaiserslautern.de/dataset/
      ↪ ortsbezirksgrenzen-kaiserslautern",
    "dataTitle": "Ortsbezirksgrenzen Kaiserslautern"
}
```

of the file is composed of the name of the associated city, the word *metadata* and the extension of the JSON format, separated by a dot. In the case of Kaiserslautern, this would be *kaiserslautern.metadata.json*. The content of this file is shown in Listing 5.1.

In the following, we discuss its components in more detail:

- `locationName` represents the name of the new city, which is Kaiserslautern in this case.
- `countryName` depicts the country – in this case Germany – to which the new city belongs.
- `annotation` stands for the category according to which the new map will be divided. In other words, the category the neighborhoods are defined by. In the context of a city and especially in the case of Kaiserslautern, districts are a suitable subdivision.
- `dataUrl` specifies the *Uniform Resource Locator* (URL) that refers to the data source, which provides the required geodata.
- `dataTitle` represents the name of the data source.

The second required file is the crucial one. A GeoJSON is needed that describes the map of the city and its district boundaries. The subdivision of Kaiserslautern is provided in various formats on the city's open data portal specified in `dataUrl`. One of them is the required GeoJSON format. The file can be downloaded free of charge and without any form of registration allowing simple integration into Click That 'Hood.

Listing 5.2 provides an extract of the GeoJSON code. We can observe that Kaiserslautern is composed of a *FeatureCollection*, where the individual city districts correspond to the represented *Features*. These districts are bounded by polygons. As explained in Section 3.3, polygons are composed of several coordinate points consisting of longitude, latitude, and altitude. The GeoJSON file provided by the open data portal divides Kaiserslautern into ten districts resulting in ten features. During the integration, it is important to ensure that the structure of the GeoJSON file is equivalent to the one

**Listing 5.2:** GeoJSON file of Kaiserslautern for Click That 'Hood.

```
{
    "type": "FeatureCollection",
    "features": [
        {
            "type": "Feature",
            "properties": {
                "name": "Dansenberg"
            },
            "geometry": {
                "type": "Polygon",
                "coordinates": [
                    [[7.7, 49.4, 0.0], [7.7, 49.4, 0.0],...]
                ]
            }
        }, ...
    ]
}
```

in Listing 5.2. In particular, it is necessary that the key in the property object is defined as `"name"`. That key is specifically important for the application because its value represents the searched district in the game. Finally, it should be noted that the name of the file has to be equal to the prefix of the metadata file. Here, it is common to use the name of the integrated city. If the metadata is added as *kaiserslautern.metadata.json*, the geofile has to be named like *kaiserslautern.geojson*.

Once Kaiserslautern is integrated, the application can be built and started locally as described in Section 5.3.1. Following that, Kaiserslautern can be selected in the game. After selecting Kaiserslautern, we get to the interface shown in Figure 5.1. Choosing *Start* then leads to the in-game view displayed in Figure 5.3. Here, the map of Kaiserslautern builds the basic layout. Initially, all districts are black and the timer is set to zero. By starting the game, the timer begins and above the map, the searched district is shown. For each properly clicked district, the corresponding area of the map turns green. Finally, the whole map is green and the time needed is displayed.

## 5.4 Future Work

Click That 'Hood has long ceased to be a small game designed to encourage learning the names of neighborhoods within a city. The game has now been expanded to include numerous categories such as countries, states, regions, and continents. In addition to these classic categories, there are also historical maps like the countries of Europe in 1914 added. Furthermore, datasets of the fastest roller coasters or airports in the world and

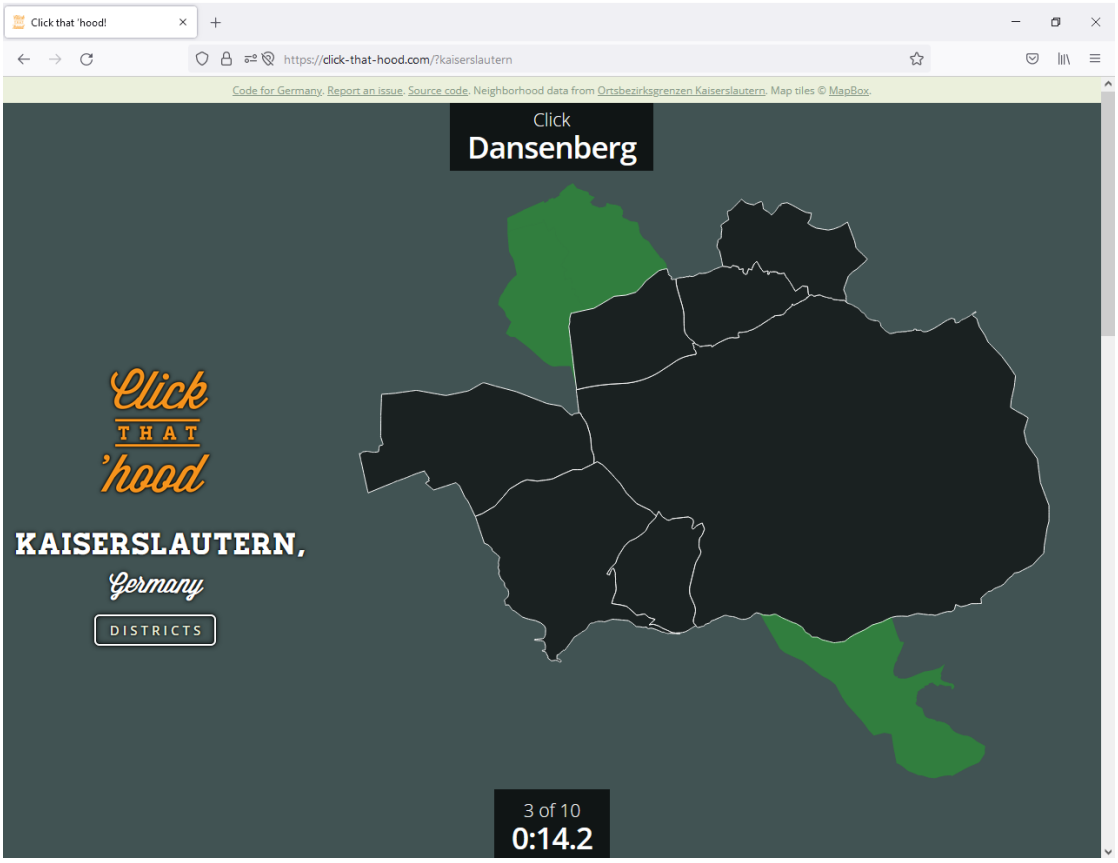



**Figure 5.3:** In-game interface of Click That 'Hood showing Kaiserslautern.

the London Tube stations are further examples that illustrate how creative developers can be when integrating new data. Moreover, the heuristic that supports slow players with tooltips, shows the expansion potential of the game. In the following, an outlook is given on how Click That 'Hood could be further extended:

**Add new cities/countries/etc.** Adding new cities is the philosophy of the game. Meanwhile, the portfolio of available cities is large, but of course, not all cities are available yet. Especially smaller cities have the opportunity to join the smart city projects through Click That 'Hood.

**Add new dimensions of categories.** We have seen that over time, unusual categories such as airports or roller coasters have been added. We can take the thought and add, for example, rivers or mountains as well. Furthermore, projected to the urban area, important city locations can also be added. An example would be bars or night clubs, that might especially attract the younger audience. But also cinemas, theaters, shopping malls or similar would be interesting. These would provide new residents with a good overview of the city's offerings, and old residents could test

their knowledge of the city. Especially for university city's like Kaiserslautern, which receive many new students every year, this is an interesting idea.

**Visualize the learning success.** A display of the required time is useful to bind players to the game for a long period of time. This allows players to follow their learning process, which provides a sense of achievement.

**Enable the comparison with other players.** As we have seen in Section 5.2.2, the gamification aspect is of great importance. To address this aspect in more detail, things like rankings and scores come to the fore. By sharing the time needed for a certain category with other players, a form of competition is created. This could be implemented by registering users, with the best ones being displayed in a high score table. Furthermore, the results of the played matches could be made shareable on social media like Facebook and others. This principle was realized by You Don't Know Africa which was introduced in Section 5.2.1.

The above is just a taste of how Click That 'Hood could further develop. From this, it becomes clear that the application can be more than just a simple game. The game can be applied in a wide variety of areas as a tool to provide people with information in a playful way.

# Chapter 6

# ParkenDD

In this chapter, we present an application named *ParkenDD*. The app displays the parking occupancy based on real-time open data. In Section 6.1, we give a motivation, in which the application is explained. Section 6.2 continues with an overview of related work focusing on smart parking and further applications. The integration of a new city into ParkenDD is presented in Section 6.3, on the example of Kaiserslautern. The chapter concludes with an outlook on possible future projects in Section 6.4.

## 6.1 Motivation

Finding an empty parking space is a common problem in rural areas. Especially in cities that one is only visiting, the search for a free spot can be frustrating. But also in our hometown, the information where public parking spaces are still available would make life much easier. According to a study[14] from 2017, Germans spend an average of 41 hours a year looking for a parking space. Besides the wasted time, this results in increased fuel consumption and environmental emissions. In the end, this leads to costs for the economy, whereby the study calculates more than 40 billion euros.

One possibility to reduce the parking search is the application ParkenDD developed by the OK Lab Dresden, which is a portmanteau of *Parken* – the German word for parking – and the city's license plate *DD*. The application is available since 2015 and can be downloaded for free on numerous platforms. This includes the mobile app for iOS and Android as well as a desktop version for Windows. The download references can be found at the ParkenDD homepage at `https://parkendd.de/index.html`.

ParkenDD shows the current parking situation in some European cities based on open data. The parking space utilization and the locations of the parking lots are visualized in real-time. In addition, for a few cities, including Dresden, a forecast of the parking space utilization for any day in the future is provided. The data required for this app is mainly provided by the respective cities. This data is then processed and delivered to the application via an interface. The interface is freely available and can be used and

[14] Study by *INRIX*: "Smart Parking – A Silver Bullet for Parking Pain" (2017), `https://inrix.com/blog/parkingsurvey/`, Graham Cookson.

adapted by anyone. As in the Click That 'Hood project, the application builds on an active community, which ensures the realization of further cities.

Currently, predominantly from Germany, ParkenDD contains twelve cities on a stable basis. Besides Dresden, also Hamburg, Freiburg, Karlsruhe, and Heidelberg are included. However, non-German cities like Basel, Zurich, and Aarhus have made a start for their country.

Within the scope of this thesis, the city of Kaiserslautern is added. In Figure 6.1, we can see two screenshots of the iOS version of ParkenDD. The screenshots show the app on the example of Kaiserslautern. Figure 6.1a depicts a list of the parking garages in Kaiserslautern. To the right of the respective parking garage and the corresponding address, the number of free spaces and the occupancy rate in percent are shown. The occupancy rate is additionally illustrated by the background color. Here, green stands for "many available parking spaces" and red for "the parking garage is (nearly) full". If we select a parking garage, the location is displayed on a map (see Figure 6.1b). In Section 6.3, the integration process of Kaiserslautern, which produces the results shown in the screenshots, is discussed in more detail.

## 6.2 Related Work

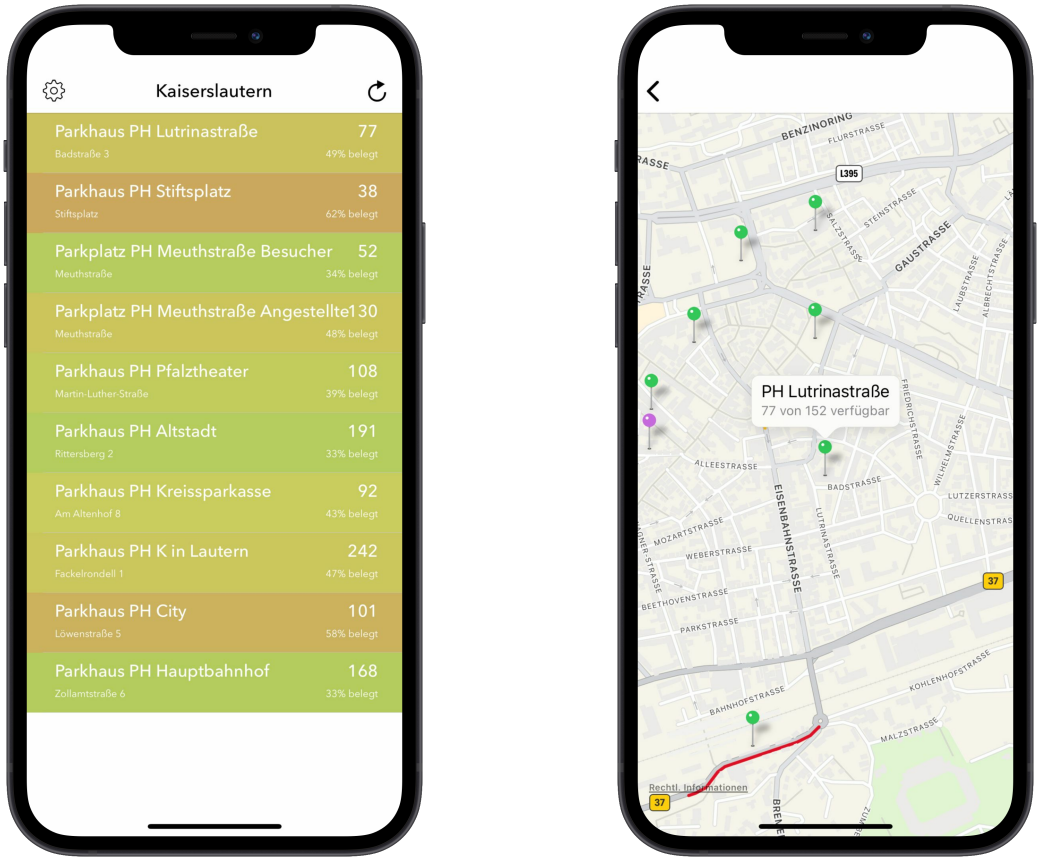
In this section, we provide an overview of topics related to ParkenDD. In Section 6.2.1, we present other popular parking applications and in Section 6.2.2, we introduce the principle of smart parking, which is an important aspect for smart cities.

### 6.2.1 Further Parking Applications

In the following, we present further applications for parking management. Firstly, we introduce the *Mobypark* project, which is an example of the urban implementation of the smart parking idea. Secondly, we introduce two popular applications *Parkopedia* and *PARK NOW*, which offer a mobile solution for parking management.

#### Mobypark

One city that has successfully realized the smart parking idea is Amsterdam. With their project Mobypark they set a good example to make parking easier and more efficient. Mobypark displays both public and private parking spaces on a platform. Thereby the project is based on the principle of *park space sharing*. The parking spaces are provided by hotels, hospitals, companies, universities, private persons, and others. The user then has the option of booking them via the portal. In addition to Amsterdam, numerous other cities such as Paris, Rotterdam, and Frankfurt are available. An instance of the platform can be found at `https://www.mobypark.com/`.



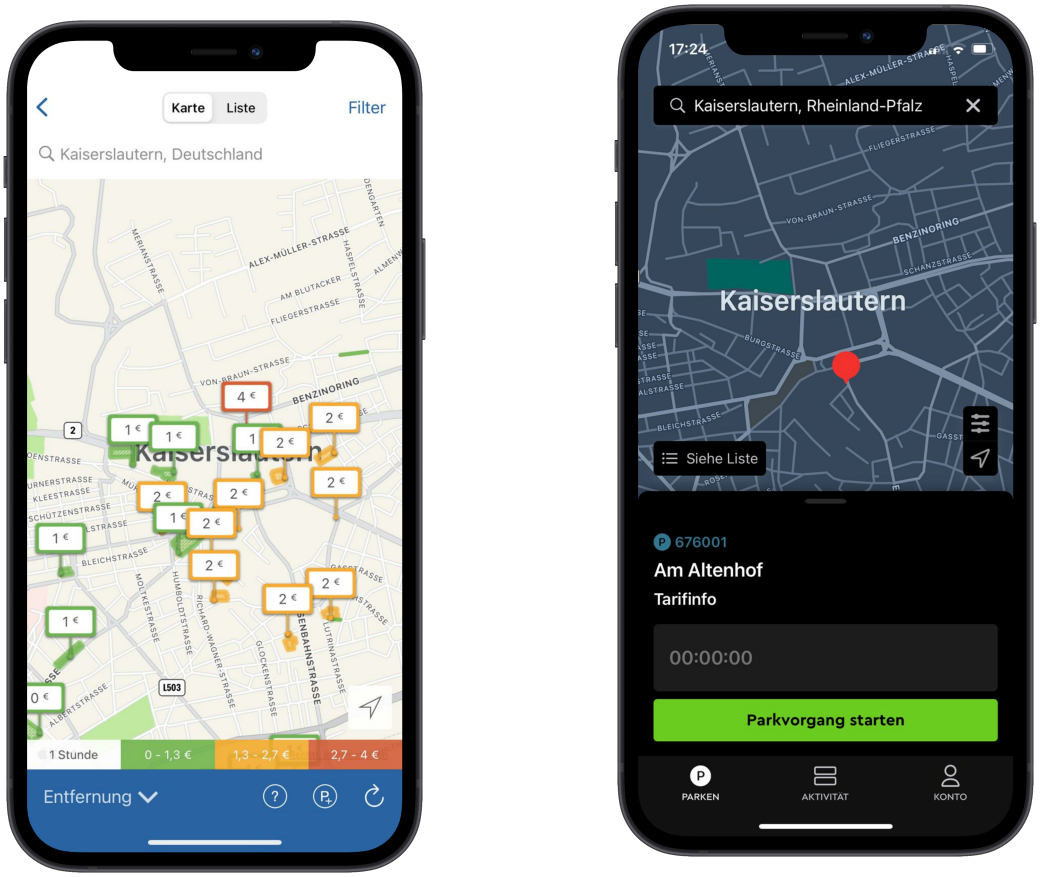

**(a)** Capacity utilization of Kaiserslautern's available parking garages.

**(b)** Location of the first parking garage after its selection.

**Figure 6.1:** Screenshots of the iOS ParkenDD app.

### Parkopedia

Parkopedia is an application developed in 2007 which displays parking information all over the world. The application is available both as a mobile app and as a browser application, which can be reached at `https://www.parkopedia.de/`. Covering over 70 million parking spaces in 89 countries, Parkopedia is the world's leading parking service provider. Like ParkenDD, Parkopedia shows parking options for a selected location. In contrast to ParkenDD, Parkopedia provides additional information, such as the price and the maximum parking duration. In addition, users can be navigated to the selected parking garage. Furthermore, Parkopedia allows users to reserve parking spaces and pay online via the app. Figure 6.2a depicts a screenshot of the application.




**(a)** Screenshot of the iOS Parkopedia app.

**(b)** Screenshot of the iOS PARK NOW app.

**Figure 6.2:** Depiction of the two introduced parking apps showing Kaiserslautern.

### PARK NOW

PARK NOW is the most popular parking application in Europe. Currently, over 330 cities in Germany and Austria are available. The PARK NOW homepage can be reached at `https://de.park-now.com/`, while the app is only available as a mobile version for iOS and Android. The app can be used to display free parking spaces in a region. Like in Parkopedia, the tickets can be paid via the app. The parking process can be started and ended manually in street parking lots. The digital ticket is checked on the basis of the car's license plate number, which enables the payment for an exact parking duration. In addition, contactless entry and exit are possible in parking garages. Figure 6.2b depicts a screenshot of the application.

### 6.2.2 Smart Parking

As the population in urban areas grows, so does the amount of traffic. In this context, smart parking solutions can significantly facilitate the search for parking lots and allow more efficient use of these. [24]. *Smart parking* can be defined as the use of technology to enable drivers to find, reserve, and pay for parking services [46].

A study by Anke and Scholle [2] provides an overview of the potential benefits resulting from smart parking, whereby these are divided into the user groups *private parking users*, *commercial parking users*, and *municipal actors*. For the private parking users, the primary benefit is the time savings and the associated cost reduction due to the shorter travel distance. Furthermore, the goal is to avoid illegal parking and to optimize parking fees by means of flexible electronic payment. The commercial users are delivery services, which benefit from the advantages of the private users too. In this case, an additional factor is the avoidance of illegal parking in the respective delivery zones. This should enable delivery traffic to be carried out more efficiently and reduce revenue losses. The third group focuses primarily on cities. Here, smart parking should reduce the volume of traffic and thus environmental emissions. In addition, user data will be used to make forecasts about future parking requests in order to enable better planning. Moreover, smart parking enables the public authorities to be guided to the wrong parking spots, which means that more wrong parkers are spotted and thus more fines are collected.

Smart parking is also targeted in Germany. Since 2016, with the *mFund* initiative, the *Bundesministerium für Verkehr und digitale Infrastruktur* (BMVI) has been funding research and development projects related to digital data-based applications for *Mobility 4.0*.[15] In this context, a study by mFund emphasizes the relevance of parking management to improve the search for parking spaces [34]. However, the study clarifies that the potential of parking management is only rarely exploited because the necessary data basis has not been created in the cities and municipalities. For this reason, mFund is demanding the nationwide publication of communal parking data and an increase in capacity to process it.

To obtain parking data, there are several techniques and methods, which we will discuss in the following. Polycarpou et al. [44] provide an overview of various proven methods for detecting available parking spaces. One method is to place sensors at each parking space to measure its status. Ultrasonic sensors can be used, that measure the distance to the parking space, which changes when a car occupies it. Furthermore, so-called robotic overhead systems or magnetic sensors can be applied to detect free parking spaces. In addition to these detection options, Polycarpou et al. mention smart parking concepts that have proven to be successful. Parking reservations eliminate the search process and prevent chaos. Dynamic price adjustments can reduce traffic. This is made possible, for example, by setting higher prices during peak hours and lowering them during off-peak hours. They also emphasize the importance of mobile solutions to find parking spaces, which further underlines the relevance of ParkenDD.

[15] Article "'mFund – Unsere Förderung für die Mobilität 4.0" (2021), `https://www.bmvi.de/DE/Themen/Digitales/mFund/Ueberblick/ueberblick.html`, BMVI.

Another interesting approach to parking space detection is given by Pala et al. [41]. *Radio Frequency Identification* (RFID) technology is used to identify vehicles and collect parking fees. RFID is a radio wave-based technology that enables the wireless exchange of data. With RFID readers, vehicles can be checked in and out without the need for personnel. In addition, fees can be collected without stopping vehicles.

A more recent and detailed overview is provided by the survey of Lin et al. [35], which covers the parking solutions from 2000 to 2016. Moreover, they introduce and evaluate the three "macro-themes" named *information collection*, *system deployment*, and *service dissemination*.

## 6.3 Integration of Kaiserslautern

In the following, we look at the integration of Kaiserslautern. Again, Kaiserslautern is just one example that can be applied to other cities. Furthermore, the technical realization of the application is only superficially discussed here because it is not the focus of this thesis. In Section 6.3.1, we first consider setting up the local environment of ParkenDD, which is required to integrate new cities. After that, in Section 6.3.2, we focus on integrating the data of Kaiserslautern to the application.

### 6.3.1 Setting up the Local Environment

ParkenDD uses an interface for the delivery of the parking lot data. This interface provides the parking information in a JSON API, which serves as a backend for the actual application. An *Application Programming Interface* (API) is an interface that serves as a standardized exchange of information between an application and individual program modules. A public API for all integrated cities is available at `https://api.parkendd.de/`. The first step is to clone the *ParkApi* project available at `https://github.com/offenesdresden/ParkAPI` and set it up using the following commands:

```
1 $ git clone https://github.com/offenesdresden/ParkAPI.git
2 $ cd ParkApi
3 $ virtualenv venv
4 $ source venv/bin/activate
5 $ pip install -e .
```

The project is mainly implemented using the Python programming language, with version 3.7 as a minimum requirement. In addition, the Python package installer *pip* and the Python tool *virtualenv* have to be installed. An introduction to Python including pip and virtualenv can be found in Section 3.5. Furthermore, a *Docker* environment is included in the project, which makes the setup process easier. Docker is free software that supports file management through container virtualization. This allows individual services and applications to be installed, updated, and launched in isolation from each other. For this reason, a Docker installation is required for the following steps.

After cloning the project, we navigate to the *ParkApi* directory in line 2. Line 3 creates a virtual environment, which is activated in line 4. Afterward, the required

**Listing 6.1:** Snippet of the docker-compose.yaml of ParkenDD.

```
1  services:
2    db:
3      image: postgres:13
4      volumes:
5        - database_data:/var/lib/postgresql/data
6      environment:
7        - POSTGRES_DB=parkapi
8        - POSTGRES_USER=parkapi
9        - POSTGRES_PASSWORD=parkapi
10     ports:
11       - 5432:5432
12   ...
```

packages can be installed using `pip install` in line 5. The next phase is to start the database connection and the server. For this purpose, the *docker-compose.yml* included in the project can be used. ParkenDD applies *PostgreSQL*[16] as database, whereby the necessary Docker image can be viewed in Listing 6.1. However, before starting with the initialization, we first need to customize the *YAML*[17] file. Listing 6.1 shows the database container from the *docker-compose.yml*. The first customization is done in line 3. Here, the version of the postgres package is set to version 13. This is done due to problems with the current version 14. Secondly, lines 10 and 11 are added so that the server connection can be found at port 5432.

Once the aforementioned changes have been made, the current images for the containers specified in *docker-compose.yml* can be downloaded and started using the commands:

```
$ docker-compose pull
$ docker-copose up -d
```

To initialize the server and database a *config.ini* file in the *ParkApi* directory is needed. For this, the example file *config_travis.ini* can be used, which must be renamed to *config.ini*. An excerpt of the utilized *config.ini* file including the configuration for the server and database connection can be found in Listing 6.2. Only line 5 is adjusted so that the database address matches the one from *docker-compose.yml*. Also, the username (*parkapi*) and password *(parkapi)* have to be included in the address.

Finally, the database can be initialized by running the Python program *parkapi-stupdb.py*. This creates a database called *parkapi*, which stores the required data of the parking spaces in the *parkapi* table. The execution of *parkapi-server.py* then starts the server, which makes the JSON API accessible via `http://localhost:5000/`.

[16]The official homepage of PostgreSQL can be accessed at `https://www.postgresql.org/`.
[17]YAML is a data serialization language, that is commonly used for file configuration.

**Listing 6.2:** Excerpt of the config.ini file of ParkenDD.

```
[DEFAULT]

port = 5000
host = ::1
database_uri = postgresql://parkapi:parkapi@localhost:5432/
    ↪ parkapi
debug = false
live_scrape = false
```

### 6.3.2 Integration of the Required Files

Real-time data is required for the indication of free parking spaces. Ideally, the city should provide this data in XML or HTML format. In the case of Kaiserslautern, this data is provided in XML format by the open data platform at `https://www.kaiserslautern.de/live_tools/pls/pls.xml`. of the city. An introduction to XML can be found in Section 3.1. Listing 6.3 shows an excerpt of the file. The individual components are discussed in the following:

- `Zeitstempel` represents the time at which the data was last updated.
- `Parkhaus` expresses each parking garage.
- `ID` is the unique identifier that is assigned to each parking garage.
- `Name` depicts the name of the parking garage.
- `Gesamt` returns the total number of all parking spaces of the parking garage.
- `Aktuell` returns the number of occupied parking spaces.
- `Trend` indicates the last tracked event i.e. if the last change was an entry (+1) or exit (-1). This information may also not be available (0).
- `Status` indicates the state of the parking garage. It is meant whether the parking garage is open/closed or has a malfunction.
- `Oeffnungszeit1` represents the opening hours of the parking garage on weekdays. However, up to now, the data for opening `Oeffnungszeit1` has not been maintained on a regular basis.
- `Oeffnungszeit2` indicates the opening hours of the parking garage on weekends.

Furthermore, a GeoJSON file is required for the parking garage locations, which was created as part of this thesis using the resource `http://geojson.io/`. By now, it can

**Listing 6.3:** Excerpt of the XML file containing the parking data of Kaiserslautern.

```
<Daten>
    <Zeitstempel>11.11.2021 18:48:00</Zeitstempel>
    <Parkhaus>
        <ID>2</ID>
        <Name>PH Lutrinastraße</Name>
        <Gesamt>137</Gesamt>
        <Aktuell>66</Aktuell>
        <Trend>0</Trend>
        <Status>Offen</Status>
        <Oeffnungszeit1>
            <von>00:00:00</von>
            <bis>00:00:00</bis>
        </Oeffnungszeit1>
        <Oeffnungszeit2>
            <von>07:00:00</von>
            <bis>20:00:00</bis>
        </Oeffnungszeit2>
    </Parkhaus>
    ...
</Daten>
```

be found in the open data portal at `https://opendata.kaiserslautern.de/dataset/parken-in-kaiserslautern` and an excerpt of the file is depicted in Listing 6.4. The excerpt shows the use of a `FeatureCollection`, which is explained in Section 3.3. Thereby, the individual features correspond to the parking garages including address and name. The sort of the parking lot is specified by `type`. In the case of Kaiserslautern, only parking garages are displayed but normal parking spaces would be conceivable too. The locations are specified by point coordinates as described in Section 3.3. The only exception is the `Feature` of the type `city`. Here, a point coordinate is given, which represents the respective city. Moreover, the key `source` contains a reference to the used dataset.

With the GeoJSON file shown in Listing 6.4 and the XML interface described in Listing 6.3, the necessary foundation of data has been created. The next step is to scrape the required data and provide it to the application in a suitable shape. The scrape function for the data of Kaiserslautern is shown in Listing 6.5. Here, the function `parse_html` takes the XML file as a parameter. In order to address the individual XML elements in an easy way, the parser `BeautifulSoup` is used. In line 4, the timestamp is extracted from the XML before it is converted to the appropriate format in line 7. Afterward, a dictionary data is created. Thereby the variable `timestamp` is linked with the key `last_updated`. Additionally, `data` contains the entry `lots`, which has an empty list as value.

**Listing 6.4:** GeoJSON file of the parking spaces for ParkenDD.

```
{   "type": "FeatureCollection",
    "features": [
    {   "type": "Feature",
        "geometry": {
            "type": "Point",
            "coordinates": [7.76, 49.44]},
        "properties": {
            "name": "Kaiserslautern",
            "type": "city",
            "source": "https://www.kaiserslautern.de/
              ↪ live_tools/pls/pls.xml"}},
    {   "type": "Feature",
        "properties": {
            "name": "PH Lutrinastraße",
            "address": "Badstraße 3",
            "type": "Parkhaus"
        },
        "geometry": {
            "type": "Point",
            "coordinates": [7.77, 49.44]}
    },  ... }
```

Starting at line 14, we use a for-loop to iterate over the individual parking garage elements of the XML file. Thereby, we extract the name, the number of occupied spaces, and the number of total parking spaces in lines 15 to 17. In line 18, the number of free parking spaces is calculated from the total amount of parking spaces excluding the number of used ones. Moreover, the state of the parking lot is stored in the variable `state`. Since ParkenDD only accepts the states *open*, *closed* and *nodata*, a translation is done in lines 20 to 26. The geodata of the GeoJSON file matching the parking garage is stored in the variable `lot` (l. 28).

In lines 29 to 38, we generate a dictionary of relevant data for the respective parking garage, which is appended to `data[lots]` for each parking lot. Free parking spaces, the total number of parking spaces, and state are taken from the XML file. Name, address, coordinates, parking type, and a unique `id` are obtained from the GeoJSON file. However, this is solely dependent on the parking data provided by the city and can be varied. The function finally returns the dictionary `data` consisting of the required parking space information.

The Python function for the scraper is stored in the file `kaiserslautern.py` of the directory `park_api/cities`. The GeoJSON is located in the same directory but has the name `kaiserslautern.geojson`. Once the preliminary work is done, we start the

**Listing 6.5:** Python scrape funtion for ParkenDD.

```
def parse_html(xml):
    soup = BeautifulSoup(xml, "html.parser")
    try:
        last_updated = soup.select("zeitstempel")[0].text
    except KeyError:
        last_updated = utc_now()
    timestamp = datetime.strptime(last_updated[0:16],
                                  "%d.%m.%Y %H:%M").isoformat()
    data = {
        "last_updated": timestamp,
        "lots": []
    }

    for ph in soup.find_all("parkhaus"):
        lot_name = ph.find("name").text
        lot_actual = int(ph.find("aktuell").text)
        lot_total = int(ph.find("gesamt").text)
        lot_free = lot_total - lot_actual

        stateGerman = ph.find("status").text
        if stateGerman == ("Offen"):
            state = "open"
        elif stateGerman == ("Geschlossen"):
            state = "closed"
        else:
            state = "nodata"

        lot = geodata.lot(lot_name)
        data["lots"].append({
            "name": lot.name,
            "free": lot_free,
            "total": lot_total,
            "address": lot.address,
            "coords": lot.coords,
            "state": state,
            "lot_type": lot.type,
            "id": lot.id
        })

    return data
```

scrapers of all cities by executing the Python file `parkapi-scraper`, which extracts the provided data and persists it in a database.

## 6.4 Future Work

ParkenDD is a prime example of a useful application based on open data. It creates a solution for the everyday parking problem by visually illustrating the parking options. This ultimately results in a reduced traffic volume. The mobile app can be accessed at any place and at any time and the simple design makes it easy to use. However, there are some extension ideas, which could lead to an improvement of the app. In the following, an outlook is given, on how ParkenDD could be further extended:

**Add new cities.** As with Click That 'Hood, the idea of the application is that more cities are added. At first, perhaps only the own hometown may be interesting. But especially when visiting a new city, the app can be useful. Here, ParkenDD first provides an overview of where parking facilities exist at all. Additionally, the presence of many cities reduces the risk of needing a secondary app since the required city is probably included.

**Provide a single comprehensive data source.** Currently, the required data is taken from two sources. The free parking spaces are taken from the provided XML interface, whereas the parking garage locations are obtained from the GeoJSON file. Here, it would be convenient to provide all the information compactly in a single interface.

**Enable the purchase of tickets.** The three presented apps of Section 6.2.1 show that the competitors offer more than just the visualization of parking capacities. Parking spaces can be reserved and tickets can be paid online in advance. By providing this functionality, the app corresponds to a complete solution for the whole parking process. The enabling of the ticket purchase would be an important aspect for ParkenDD in order to compare itself with existing providers.

**Provide parking space sharing.** To expand the parking supply, ParkenDD could be expanded to include parking space sharing. As with the Dutch pioneer Mobypark from Section 6.2.1, municipal facilities and private individuals could offer their parking spaces for rent.

**Predict future parking occupancy.** ParkenDD offers the prediction of future parking garage occupancy for the city of Dresden. This functionality could be provided for other cities.

**Display of additional parking information.** By now, only the name, address, location, and occupancy rate of the parking facility are shown. However, the fee cost may also be of interest. Additionally, a general overview of the opening hours and any contact information would be helpful. Furthermore, additional information

such as the availability of disabled and women's parking spaces and the maximum vehicle measurement would be useful.

The aspects mentioned above illustrate the potential of ParkenDD, which offers the possibility for larger projects in the future. The implementation of these aspects would enable competition with other providers.

# Chapter 7

# Mietmap

In this chapter, we present an application named *Mietmap*, which visualizes the rental apartment prices of the city of Karlsruhe in 2015. In Section 7.1, we give a motivation, in which the application is explained. Section 7.2 continues with an overview of related work focusing on web scraping. The integration of a new city into Mietmap is presented in Section 7.3, on the example of Kaiserslautern. The chapter concludes with an outlook on possible future projects in Section 7.4.

## 7.1 Motivation

In Germany, the rental prices of apartments have been rising steadily over the past few years. This is the result of an analysis by the online platform *ImmoScout24*,[18] which evaluated 8.5 million real estate advertisements over the last five years. For this reason, it has become more important to compare the offers in order to find the best deal. Additionally, a rent index is useful for apartment providers in order to be able to classify their own apartments.

For this purpose, in 2015, the OK Lab Karlsruhe has developed a visualization of the rental prices of their city. The visualization named *Mietmap* – whereby *Miet* is the German word for rent – corresponds to an interactive web application, which can be accessed at `https://codeforkarlsruhe.github.io/mietmap/`. In Figure 7.1, we present a screenshot of the visualization based on the data of Kaiserslautern. The individual rental offers are shown as points on the map of Kaiserslautern. The respective offer prices are illustrated by the background color of the point, whereby the classification can be taken from the legend in the lower right edge of the picture. To obtain the apartment data, we use a scraper that extracts the apartment listings from the real estate portal ImmoScout24 and converts them into the appropriate format. In the end,

[18] Blog post "ImmoScout24 WohnBarometer: Angebotsmieten entwickelten sich moderat" (2021), `https://www.immobilienscout24.de/unternehmen/news-medien/news/default-title/immoscout24-wohnbarometer-angebotsmieten-entwickelten-sich-im-dritten-quartal-moderat-am-deutlichsten-zogen-die-preise-in-leipzig-erfurt-hamburg-und-muenchen-an/`, ImmoScout24.

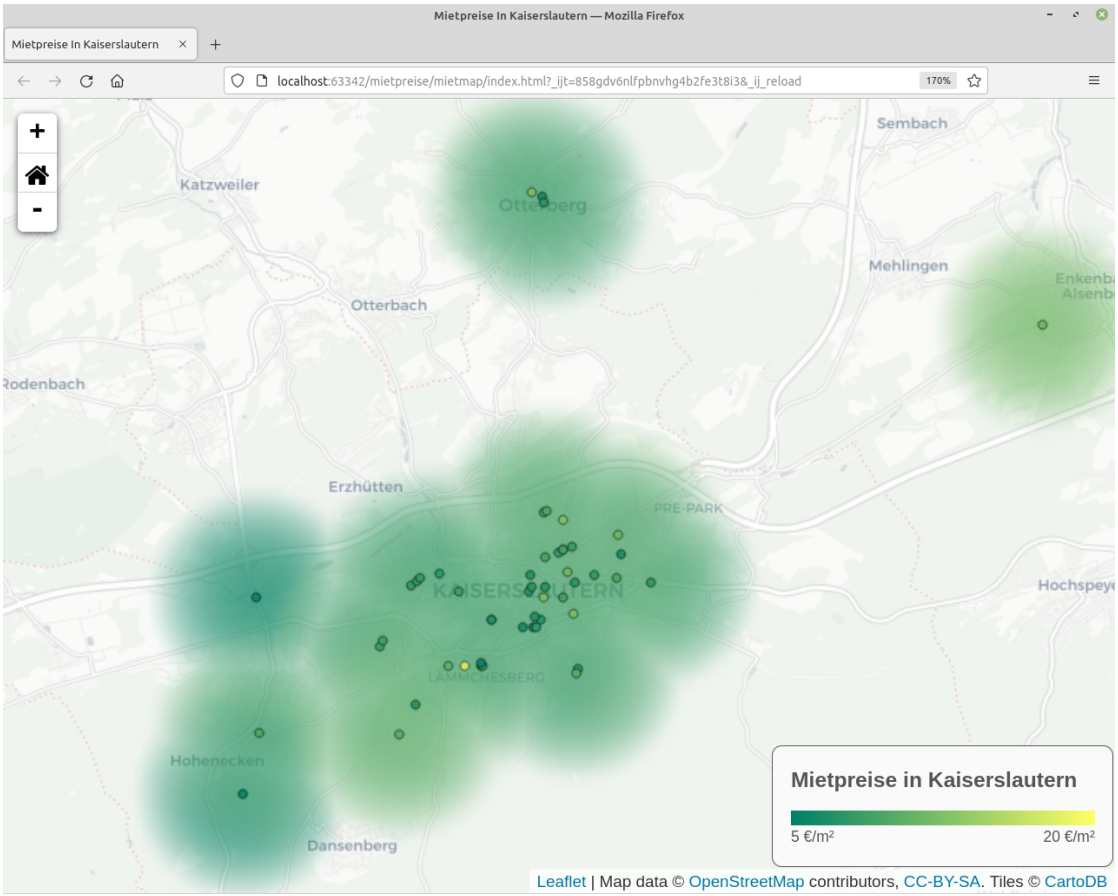



**Figure 7.1:** Mietmap visualization of the rental prices of Kaiserslautern.

a clustering algorithm is used to generate a heatmap that visually highlights the price distribution.

In this thesis, the visualization is applied to the city of Kaiserslautern. The idea of a rent index for the orientation of rental prices is not new to Kaiserslautern. The city's rent index is available at `https://www.kaiserslautern.de/mb/themen/statistik/pdf/2021mietspiegel.pdf`. However, the provided data cannot be used meaningfully for the presented rent map. In addition, the city's rent index is not collected according to scientifically accepted principles. Furthermore, only a few criteria are recorded, so that the overview of the actual offers obtained from a real estate portal probably reflects the more realistic situation. The integration of Kaiserslautern, and the associated data collection, is shown in Section 7.3.

## 7.2 Related Work

In this section, we provide an overview of topics related to Mietmap. In Section 7.2.1, we present other interactive web visualizations developed as part of the CFG project. In Section 7.2.2, we introduce the technique of *Web Scraping*, which is used to get the required rental data.

### 7.2.1 Further Interactive Web Visualizations

In the following, we present two further interactive web visualizations. Firstly, we introduce a price map for rail and bus travel in Europe. Secondly, we present an application called *Wo ist Markt?* that visualizes the weekly city markets for multiple cities.

#### Train and Bus Fare Map

The OK Lab Berlin is currently working on an interactive price map for rail and bus travel in Europe. Based on a world map, it displays the cheapest connections to many European cities for the European area. Thereby, the individual offers are displayed together with their prices. By selecting an offer, the user is redirected to the corresponding website for ticket purchase. A first instance of the project can be found at `https://pricemap.eu/?origin=DEBER`.

#### Wo ist Markt?

In 2015, the OK Lab Karlsruhe developed an interactive visualization of the weekly markets in their city. The project follows the same principle as Click That 'Hood and ParkenDD, which were introduced in Chapters 5 and 6. New cities can be added to the visualization by integrating a JSON file including the required data. By now, a large number of German cities – including the city of Kaiserslautern – are available in the visualization. The visualization of Kaiserslautern can be accessed at `https://www.wo-ist-markt.de/#kaiserslautern`. More information on this application can be found in Chapter 4.

### 7.2.2 Web Scraping

*Web scraping* (also known as *web harvesting* or *screen scraping*) is a data mining technique for the extraction of unstructured data from websites [49]. The extracted data can then be persisted in a database in order to allow further analysis and processing. Here, various techniques exist to obtain the desired information.

A survey by Saurkar et al. [48] introduces various web scraping techniques, which are presented in the following. The simplest method is the manual *copy and paste* of the selected content. Even though this is the best method, it is quite impractical when analyzing large amounts of data. This is where *Hypertext Transfer Protocol* (HTTP) programming can help. HTTP requests can be used to extract information from a static and dynamic web page. Furthermore, two parsing methods are presented. Either

HTML pages are parsed, or specific web page contents are accessed by using the DOM tree. Moreover, Saurkar et al. introduce the technique of using web scraping software that extracts and stores the content automatically. Finally, they mention the use of so-called *computer vision web-page analyzers*. These try to identify and extract the web page information using machine learning and computer vision approaches.

Web scraping is an approach that is used to support various areas of life. In a report by Diouf et al. [21], various state-of-the-art application areas are presented. These include job search and recommender systems. Furthermore, they mention the sectors advertisement and health, in which some topic-specific applications rely on web scraping. Above all, they emphasize the area of journalism. Here, web scrapers support the daily task of information extraction in an automated manner.

## 7.3 Integration of Kaiserslautern

In the following, we look at the integration of Kaiserslautern. Here, Kaiserslautern is just one example that can be applied to any other city. Furthermore, Mietmap consists of some more complex techniques, which are only superficially addressed in this thesis. In Section 7.3.1, we first consider setting up the local environment, which is required to integrate new cities. After that, in Section 7.3.2, we focus on integrating the necessary data for the heatmap of Kaiserslautern.

### 7.3.1 Setting up the Local Environment

The rental map is composed of three GitHub projects, which are needed for the final representation. The main project is available at `https://github.com/CodeforKarlsruhe/mietmap` and can be cloned using the command:

```
$ git clone https://github.com/CodeforKarlsruhe/mietmap.git
```

The project mainly consists of JavaScript code, while the web interface is realized by HTML and CSS. A brief introduction to the concepts of JavaScript and the relationship to HTML and CSS can be found in Section 3.4.

For the extraction of the data from ImmoScout24, we need a scraper, which can be accessed via the repository at `https://github.com/CodeforKarlsruhe/mietmap-scraper` and has to be cloned like the previous repository. As with ParkenDD, the scraper is a program written in Python. Again, the Python package installer *pip* and the Python tool *virtualenv* are required, whereby an introduction to Python including pip and virtualenv can be found in Section 3.5. By applying the commands:

```
1 $ cd mietmap-scraper
2 $ virtualenv venv
3 $ source venv/bin/activate
4 $ pip install -r requirements.txt
```

we can navigate to the project directory and create a virtual environment. The virtual environment is then activated by the command presented in line 3. Finally, line 4 shows

the installation of the requirements. Here, the package *wsgiref* can be removed from the *requirements.txt* because it is included as a default library since Python 3.

For the creation of the heatmap, we need a third project, which is available at `https://github.com/CodeforKarlsruhe/mietmap-ove` and must be cloned like the previous projects.

### 7.3.2 Integration of the Required Files

The visualization is based on data of the offered apartments. A scraper is used to extract the data from a real estate platform. In the implementation from 2015, the German real estate market leader ImmoScout24 was used, which can be reached at `https://www.immobilienscout24.de/`. For this purpose, the target URL referencing the offers of Karlsruhe was opened using the package *urllib2*. By now this realization is no longer possible. ImmoScout24 uses the so-called *Completely Automated Public Turing test to tell Computers and Humans Apart* (CAPTCHA) after a few calls to avoid bot requests. Bots are computer programs that perform tasks automatically which applies to the scraper. Thus, simple integration with just changing the target URLs is no longer possible. For this reason, the *Selenium WebDriver* is utilized to open the specific URLs, whereby the CAPTCHAs are solved manually for each page. The procedure of the scraper is explained in the following.

The initial ImmoScout24 website with the offers of Kaiserslautern can be reached at `https://www.immobilienscout24.de/Suche/de/rheinland-pfalz/kaiserslautern/wohnung-mieten?pagenumber=1`. Starting with this page, all result pages are iterated. First, we extract the relevant information for each page by the use of the simplified scrape function shown in Listing 7.1. Here, the function `extract_listings` takes the parsed result page, consisting of the apartment information as a parameter. Due to the previous parsing with `BeautifulSoup`, the elements of the DOM (cf. Section 3.4) can be addressed in a simplified way. In the following, when we talk about an *element*, the element of the DOM is meant. In line 5, we iterate over the listing elements of the result page, whereby these listing elements correspond to the individual apartments. For each apartment, a unique *id* is stored in the variable `listing_id` (l. 8). In line 10, the extracted street name is stored in the variable `street_raw`. The street name is then subdivided into the `street`, `number`, and `suburb` in line 12. From line 14, we iterate over the description list elements of the `font-highlight` class. Depending on the ending, the rental price or the living space is extracted. In lines 25 to 31, the resulting dictionary is shown, which specifies a particular apartment by using the unique `listing_id` and associated information. Based on the previous calculations, the `street`, house `number`, `suburb`, `rent`, and living `area` are saved for each apartment. Finally, the resulting dictionary is stored in a simple *SQLite* database.

In the next step, we search for the locations of the extracted apartments. For this purpose, the apartments are read from the database and the locations are determined using the geocoding software *Nominatim*. For this, the included function `_geolocator.geocode(address)` returns the latitude and longitude of the given address, which are persisted in the database.

**Listing 7.1:** Python scraper function for Mietmap.

```
def extract_listings(soup):
  listings = {}
  offers = soup.find_all('li',
                         class_='result-list__listing')
  for li in offers:
    for a in li.find_all('a'):
      if a.get('href', '').startswith('/expose/'):
        listing_id = a.get('href').split('/')[-1]
        break
    street_raw= li.find('button',
                class_='result-list-entry__map-link').text
    street, number, suburb = parse_address(street_raw)
    count = 0
    for dd in li.find_all('dd', class_='font-highlight'):
      if count <= 1:
        content = dd.string.strip()
        if content.endswith('€'):
          rent = parse_german_float(content.split()[0])
        elif content.endswith('m²'):
          area = parse_german_float(content.split()[0])
        count = count + 1
      else:
        continue

      listings[listing_id] = {
        'street': street,
        'number': number,
        'suburb': suburb,
        'rent': rent,
        'area': area
      }
  return listings
```

In the last step, we store the required data in a JSON file called *marker.json*. A small excerpt of this file is shown in the following:

```
[[49.44418,7.79725,8.0],[49.4386,7.77166,7.5], ...]
```

The JSON file consists of an array, which contains several further arrays. The individual arrays correspond to the respective apartments, which are represented by latitude (e.g. 49.44), longitude (e.g. 7.79), and the price per square meter (e.g. 8.0). Price per square meter is calculated by dividing the price by the living area.

Once we got the individual data points in the appropriate JSON format, we generate an image overlay of the heatmap. This is done by the Python file *create_overlay.py*, which generates an *overlay.png* using a clustering algorithm. The resulting overlay is then stored in the main directory *mietmap*. In order to generate the heatmap for the city of Kaiserslautern, the respective area must be delimited, which is done with the help of geographical coordinates. To define a bounding rectangle, we use the upper left point and the lower right point of the rectangle. For the specification, we define the variable *HEATMAP_AREA*, which consists of two coordinates:

```
HEATMAP_AREA = ((7.635419,49.486648),(7.863928, 49.392256))
```

whereby the coordinates are represented by longitude and latitude. In order to set the *overlay.png* correctly, we specify the bounding coordinates in the *index.html* by assigning the JavaScript variable *OVERLAY_BOUNDS*. In addition, the radius for the clustering procedure, which defines the size of the resulting cluster, has to be specified in the *HEATMAP_RADIUS* variable. Depending on how much data is available, the radius can be adjusted. Accordingly, a smaller radius is useful for a small amount of data, while a larger radius can be selected for a large amount of data. In Figure 7.1, we selected a radius of 0.0005.

Figure 7.1 illustrates some of the problems associated with integrating new cities, which we will discuss in the following. The visualization of Kaiserslautern shows only a few data points. At the time of implementation, ImmoScout24 contained 179 apartment listings in the area of Kaiserslautern. However, 72 of these listings do not include an address so that it is not possible to determine the coordinates. Furthermore, in a few cases, the address is not parsed correctly due to a special format, so that the location cannot be determined. Another reason for the lack of data points is the geocoding software Nominatim, which cannot find the appropriate coordinates for all addresses. For this reason, about half of the available listings could not be used for the rental map, which results in the rather sparse representation of Figure 7.1. A prospect of future work to solve this problem is given in Section 7.4.

## 7.4 Future Work

As explained at the end of Section 7.3.2, a lot of data is needed for the creation of a meaningful visualization. In Figure 7.2, the rent map is enriched with data from all German states. For this purpose, we used the scraper presented in Section 7.3.2 to collect

and merge the apartment offers of all German states. To get a better representation, only the heatmap without the associated data points is shown in this example. The visualization provides a good overview of the rent distribution in Germany and confirms the assumption that especially large and popular cities like Berlin, Hamburg, Munich, and Frankfurt have the highest rents. This becomes visible by the individual clusters colored in yellow. Figure 7.2 demonstrates which meaningful visualizations are possible with sufficient data. In the following, an outlook on future work is given, addressing this problem among other possible extensions:

**Provide more data.** There are several ways to do this. On the one hand, the support of the respective city is helpful. By collecting and providing the apartment offers, the portfolio of data can be expanded. In addition, the scraper can be improved. Through a smarter parser, more addresses could be used for the coordinate search. Another aspect is the coordinate lookup. Here, either the coordinates could be collected manually, or another tool as Nominatim could be explored or developed.

**Present the price development.** So far, the apartment offers are collected at a certain point in time. It would be possible to crawl the apartment offers at regular intervals over a longer period of time. A time regulator could be used to interactively display the development of prices. In addition, the collected data can be utilized to create forecasts for the future market.

**Visualize other data.** In Mietmap, the rental prices of apartments per square meter are visualized. The first thought would be to apply this visualization to houses. In principle, any form of goods or services could be visualized, as long as the necessary data is provided. Thus, questions such as *where are kindergarten places still available?*, or *where is the cheapest restaurant?* could be realized. Not only prices but also ratings could be displayed. Instead of the cheapest restaurants, the best-rated ones could be displayed, for example, by using Google reviews as a source of ratings.

**Automate the CAPTCHA solving.** In our implementation, CAPTCHAs have to be solved manually. It would be useful to integrate an automated CAPTCHA solver. On the other hand, an extension of the scraper would be conceivable, so that no CAPTCHAs have to be solved at all.

The above-mentioned aspects show that the rental map visualization can be used for many topics. Due to the colored heatmap, the application manages to provide a good overview. The potential has also been recognized by the cities of Munich and Giessen, which are currently implementing an interactive rent index for their cities. The respective project descriptions can be found from CFG's project archive at `https://www.codefor.de/projekte/archiv/`.

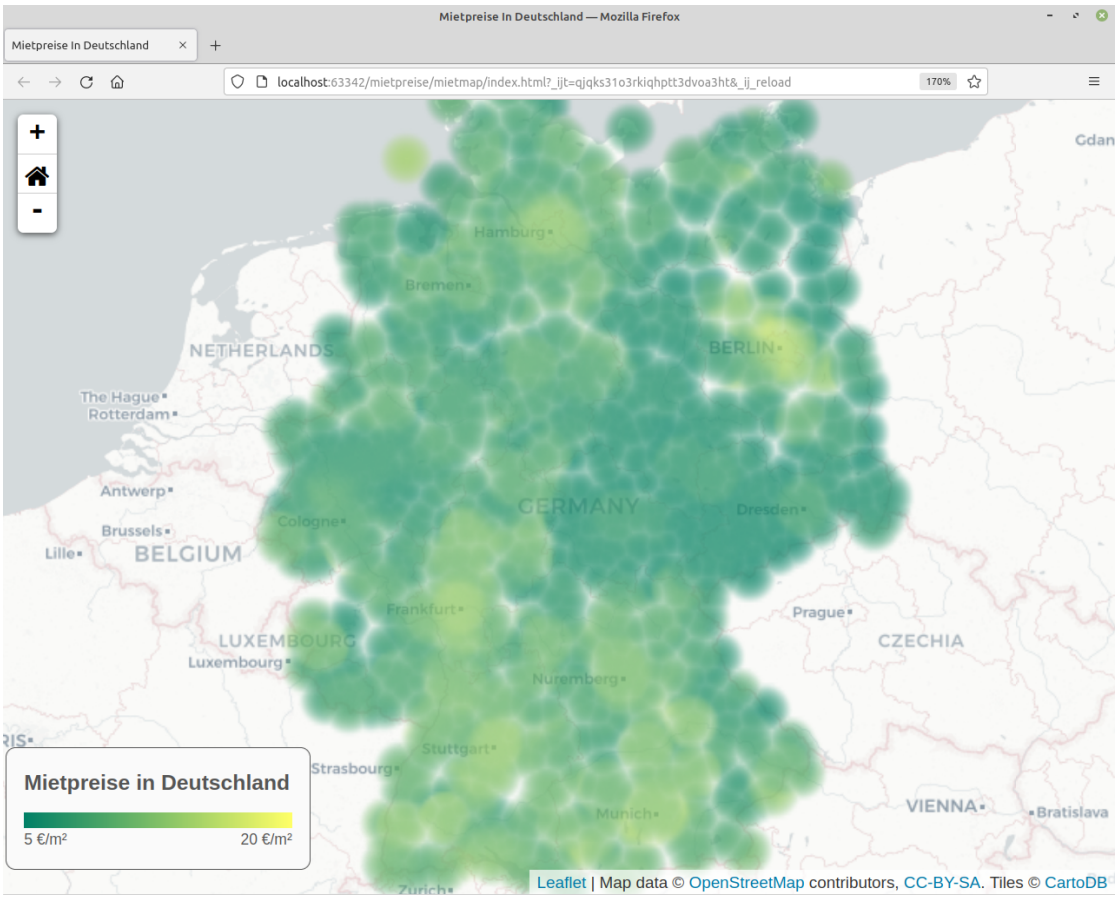



**Figure 7.2:** Mietmap visualization of the rental prices of Germany.

# Chapter 8

# Conclusion

In this chapter, we conclude the results of this thesis.

## 8.1 Summary

The goal of the thesis was to give a survey on community-driven ready-to-go applications, which can be useful in urban areas. Our focus laid on the data needed for these applications and its integration. Finally, we desired to define a minimum dataset that is essential for the individual presented applications.

In order to achieve this goal, we first introduced the underlying concepts in Chapter 2 to understand the background of the thesis. Among other things, we explained the concept of the smart city and its implementation using the example of Kaiserslautern. The importance of the community was discussed in particular, whereby the OKF including the CFG projects represents an important representative.

Since different technologies were necessary to implement the presented projects, we discussed different used technologies in Chapter 3. Besides data formats like XML, JSON, and GeoJSON, which are frequently applied in the projects for the provision of data, we gave an introduction to the utilized programming languages JavaScript and Python.

After all, in Chapter 4, we presented some existing ready-to-go applications, which are operated by CFG and its community. The criterion was that the projects relate to the urban area, and are suitable for redeployments. Furthermore, we highlighted which data should be available for redeployment, how much effort is required for it, and how many adaptations already exist.

To demonstrate such an adaptation process, we selected three projects, which have been set up and supplied with the data from the city of Kaiserslautern. For each project, we individually addressed related work and gave an outlook on future work. The first project Click That 'Hood was presented in Chapter 5. Click That 'Hood is a geography game, which has already been implemented for many cities. For the adaption of Kaiserslautern, we integrated the city's district boundaries in GeoJSON format into the application.

In Chapter 6, we presented the application ParkenDD, which should support the user in finding a free parking space. Here, the available parking lots were provided via a live-API in XML format. Using a scraper written in Python, we extracted the required data and converted it to the desired format. For the display of the parking lot locations, we manually created a GeoJSON file consisting of the individual lots, which was finally adopted as an official dataset by the open data portal of Kaiserslautern.

In Chapter 7, we introduced the application named Mietmap, which visualizes the city's rental prices on a heatmap. We extracted the required data for Kaiserslautern from the real estate platform ImmoScout24 by the use of a scraper. In this case, we were particularly impressed by the difficulty of obtaining the data. Disturbing factors were the blocking of scrape bots from ImmoScout24, the lack of data on the platform, and the challenging parsing of the addresses for location determination. This project therefore serves as a good example of data that should be officially provided as open data by municipalities instead. The extracted data was finally passed in JSON format to a clustering algorithm that generated the resulting heatmap.

## 8.2 Discussion

We observed that the publication of urban data is a crucial aspect of the establishment of smart cities. It allows more applications to be developed that simplify urban life. The most important stakeholders, namely the citizens, thus have more opportunities to get involved in urban management and development. That is the reason why smart cities and smart projects are being funded in Germany.

And the funding seems to be successful. We noticed that currently in Germany several projects exist that are beneficial in different areas. Based on our survey we presented in Chapter 4 and especially with respect to the three projects mentioned in Chapters 5 to 7, it becomes clear that in some cases only a few data and actions are necessary to implement such projects for individual cities. In particular, data in GeoJSON format is frequently used to map certain areas, such as district boundaries or points of interest. Furthermore, in order to obtain data that is not yet available, scrapers are applied to extract them from any source. In addition, we would like to emphasize the utilization of live-APIs, which serve as an up-to-date source of information.

In summary, projects with great benefits can be developed with just a few resources. Nevertheless, in Section 2.1.3, we observed that Germany is rather in the middle of the ranking in an international comparison. In this context, the goal should be to increase the portfolio of provided data and developed applications over the next years. To this end, this thesis can be used as inspiration for further projects.

# Bibliography


[1] Jan Ackermann, Torsten Fisch, and Katharina Sauter. Kommunales Handlungsprogramm Mobilität. 2021.

[2] Jürgen Anke and Julia Scholle. Nutzenpotenziale von Smart Parking. *Digitale Transformation: Methoden, Kompetenzen und Technologien für die Verwaltung*, pages 175–187, 2016.

[3] Jr. Brent W. Benson. Javascript. *ACM SIGPLAN Notices*, 34(4):25–27, 1999.

[4] Geert Jan Bex, Frank Neven, and Jan Van den Bussche. DTDs versus XML Schema: A Practical Study. In *Proceedings of the 7th International Workshop on the Web and Databases (WebDB)*, pages 79–84, 2004.

[5] Tim Bray. The JavaScript Object Notation (JSON) Data Interchange Format. *RFC8259*, 2017. `https://datatracker.ietf.org/doc/html/rfc8259`. [Online; accessed 2021-12-01].

[6] Tim Bray, Jean Paoli, and C. M. Sperberg-McQueen. Extensible Markup Language (XML) 1.0. `https://www.w3.org/TR/1998/REC-xml-19980210`, 1998. [Online; accessed 2021-12-01].

[7] Gerard Briscoe and Catherine Mulligan. Digital Innovation: The Hackathon Phenomenon. `https://qmro.qmul.ac.uk/xmlui/bitstream/handle/123456789/11418/Briscoe%20Digital%20Innovation:%20The%20Hackathon%20Phenomenon%202014%20Published.pdf?sequence=2`, 2014. [Online; accessed 2021-12-01].

[8] Lina Bruns, Benjamin Dittwald, and Fritz Meiners. Leitfaden Für Qualitativ Hochwertige Daten Und Metadaten. Technical report, Fraunhofer-Institut für Offene Kommunikationssysteme FOKUS, 2019. [Online; accessed 2021-12-01].

[9] Peter Bühler, Patrick Schlaich, and Dominik Sinner. *Webtechnologien: JavaScript – PHP –Datenbank*. Springer, 2018.

[10] Bundesinstitut für Bau-, Stadt-, und Raumforschung (BBSR). Smart City Charta–Digitale Transformation in den Kommunen nachhaltig gestalten. `https:`

//www.smart-city-dialog.de/wp-content/uploads/2021/04/2021_Smart-City-Charta.pdf, 2017. [Online; accessed 2021-12-01].

[11] Bundesministerium des Innern, für Bau und Heimat. Smart City Dialog 2019. https://www.smart-city-dialog.de/wp-content/uploads/2020/06/smart-city-dialog-2019.pdf, 2019. [Online; accessed 2021-12-01].

[12] Bundesministerium des Innern, für Bau und Heimat. Smart Cities: Stadtentwicklung im digitalen Zeitalter. https://www.bmi.bund.de/DE/themen/bauen-wohnen/stadt-wohnen/stadtentwicklung/smart-cities/smart-cities-node.html, 2020. [Online; accessed 2021-12-01].

[13] Bundesministerium des Innern, für Bau und Heimat. Open-Data-Strategie der Bundesregierung. https://www.bundesregierung.de/resource/blob/992814/1940386/1d269a2ad1b6346fcf60663bdea9c9f8/2021-07-07-open-data-strategie-data.pdf?download=1, 2021. [Online; accessed 2021-12-01].

[14] Howard Butler, Martin Daly, Allan Doyle, Sean Gillies, Stefan Hagen, Tim Schaub, et al. The GeoJSON Format. *RFC7946, Internet Engineering Task Force (IETF)*, 2016. https://datatracker.ietf.org/doc/html/rfc7946#page-4. [Online; accessed 2021-12-01].

[15] Hafedh Chourabi, Taewoo Nam, Shawn Walker, J. Ramon Gil-Garcia, Sehl Mellouli, Karine Nahon, Theresa A. Pardo, and Hans Jochen Scholl. Understanding Smart Cities: An integrative framework. In *Proceedings of the 45th Hawaii International Conference on System Sciences*, pages 2289–2297, 2012.

[16] James Clark and Makoto Murata. RELAX NG Specification. https://www.oasis-open.org/committees/relax-ng/spec-20011203.html, 2001. [Online; accessed 2021-12-01].

[17] Code for Germany. Code for Germany – digitales Engagement. https://codefor.de/ressourcen/Flyer_CodeforGermany.pdf, . [Online; accessed 2021-12-01].

[18] Code for Germany. Code A Difference. Nutz‘ deine Fähigkeiten, um deine Stadt zu verbessern! https://codefor.de/ressourcen/Slidedeck_Was-ist-Code-for-Germany.pdf, . [Online; accessed 2021-12-01].

[19] Douglas Crockford. The application/json Media Type for Javascript Object Notation (JSON). *RFC 4627*, 2006. https://datatracker.ietf.org/doc/html/rfc4627. [Online; accessed 2021-12-01].

[20] Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. From Game Design Elements to Gamefulness: Defining “Gamification”. In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, pages 9–15, 2011.

[21] Rabiyatou Diouf, Edouard Ngor Sarr, Ousmane Sall, Babiga Birregah, Mamadou Bousso, and Sény Ndiaye Mbaye. Web Scraping: State-of-the-Art and Areas of Application. In *Proceedings of the IEEE International Conference on Big Data (Big Data)*, pages 6040–6042. IEEE, 2019.

[22] Ecma International. ECMAScript® 2021 language specification. *ECMA-262*, 2021. `https://www.ecma-international.org/publications-and-standards/standards/ecma-262/`. [Online; accessed 2021-12-01].

[23] Führungsakademie Baden-Württemberg. Smart BW – Stark in der Stadt und in der Fläche, 2015.

[24] Barbara Flügge. *Smart Mobility*. Springer, 2016.

[25] Open Knowledge Foundation. Open Definition: Defining Open in Open Data, Open Content and Open Knowledge. `https://opendefinition.org/od/2.1/en/`, 2015. [Online; accessed 2021-12-01].

[26] Rudolf Giffinger, Nataša Pichler-Milanović, Christian Fertner, Hans Kramar, Robert Kalasek, and Evert Meijers. *Smart cities: Ranking of European medium-sized cities*. Centre of Regional Science, Vienna University of Technology, 2007.

[27] GovData. GovData – Das Datenportal für Deutschland. `https://www.govdata.de/`. [Online; accessed 2021-12-01].

[28] Colin Harrison, Barbara Eckman, Rick Hamilton, Perry Hartswick, Jayant Kalagnanam, Jurij Paraszczak, and Peter Williams. Foundations for Smarter Cities. *IBM Journal of Research and Development*, 54(4):1–16, 2010.

[29] Hildebrand, Anna and Jakopin, Nejc and Reinhard, Philipp and Riegel, Lars. Der Smart-City-Markt in Deutschland, 2021.

[30] Rick Jelliffe. The Schematron Assertion Language 1.5. Technical report, Technical report, GeoTempo Inc, 2000.

[31] Simon Holm Jensen, Anders Møller, and Peter Thiemann. Type Analysis for JavaScript. In *International Static Analysis Symposium*. Springer, 2009.

[32] Raman Kazhamiakin, Annapaola Marconi, Mirko Perillo, Marco Pistore, Giuseppe Valetto, Luca Piras, Francesco Avesani, and Nicola Perri. Using Gamification to Incentivize Sustainable Urban Mobility. In *Proceedings of the 2015 IEEE First International Smart Cities Conference (ISC2)*, pages 1–6. IEEE, 2015.

[33] KL.digital. Unser Lautern – herzlich digital. `https://www.herzlich-digital.de/`. [Online; accessed 2021-12-01].

[34] Andrea Liebe and Annette Hillebrand. Platz da?! Datenbasierte Systeme zur Parkplatzerkennung. https://www.wik.org/fileadmin/mFUND_VF/mFUND-WIK-Studie_DatenbasierteSystemeZurParkplatzerkennung.pdf. [Online; accessed 2021-12-01].

[35] Trista Lin, Hervé Rivano, and Frédéric Le Mouël. A Survey of Smart Parking Solutions. *IEEE Transactions on Intelligent Transportation Systems*, 18(12):3229–3253, 2017.

[36] Marcin Wchary. Two years of clicking 'hoods. https://mwichary.medium.com/two-years-of-clicking-hoods-890e67553165. [Online; accessed 2021-12-01].

[37] Andreas Meier and Edy Portmann. *Smart City–Strategie, Governance und Projekte*. Springer, 2017.

[38] Fatima Neves, Miguel Neto, and Manuela Aparicio. The impacts of open data initiatives on smart cities: A framework for evaluation and monitoring. *Cities*, 106, 2020.

[39] Nurzhan Nurseitov, Michael Paulson, Randall Reynolds, and Clemente Izurieta. Comparison of JSON and XML Data Interchange Formats: A Case Study. *Caine*, 9:157–162, 2009.

[40] Antonio Opromolla, Valentina Volpi, Andrea Ingrosso, and Carlo Maria Medaglia. Co-design Practice in a Smart City Context Through the Gamification Approach: A Survey About the Most Suitable Applications. In *Proceedings of the International Conference on Distributed, Ambient, and Pervasive Interactions*, pages 578–589. Springer, 2015.

[41] Zeydin Pala and Nihat Inanc. Smart Parking Applications Using RFID Technology. In *Proceedings of 1st Annual RFID Eurasia*, 2007.

[42] Partnerschaft Deutschland (PD). Datensouveränität in der Smart City. https://www.pd-g.de/assets/Presse/Fachpresse/200213_PD-Impulse_Datensouveraenitaet_Smart_City.pdf, 2020. [Online; accessed 2021-12-01].

[43] Felipe Pezoa, Juan L. Reutter, Fernando Suarez, Martín Ugarte, and Domagoj Vrgoč. Foundations of JSON Schema. In *Proceedings of the 25th International Conference on World Wide Web*, pages 263–273, 2016.

[44] Elena Polycarpou, Lambros Lambrinos, and Eftychios Protopapadakis. Smart parking solutions for urban areas. In *Proceedings of the IEEE 14th International Symposium on "A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, pages 1–6. IEEE, 2013.

[45] Frank Robinson. A Proven Methodology to Maximize Return on Risk. http://www.syncdev.com/minimum-viable-product/, 2001. [Online; accessed 2021-12-01].

[46] Caroline J. Rodier and Susan A. Shaheen. Transit-based smart parking: An evaluation of the San Francisco Bay area field test. *Transportation Research Part C: Emerging Technologies*, 18(2):225–233, 2010.

[47] Dieter Rombach. Leitbild "herzlich digitale Stadt". `https://www.herzlich-digital.de/wp-content/uploads/2019/05/Leitbild_herzlich_digitale_Stadt_Kaiserslautern.pdf`, 2018. [Online; accessed 2021-12-01].

[48] Anand V Saurkar, Kedar G Pathare, and Shweta A Gode. An Overview On Web Scraping Techniques And Tools. *International Journal on Future Revolution in Computer Science & Communication Engineering*, 4(4):363–367, 2018.

[49] De S Sirisuriya. A Comparative Study on Web Scraping. In *Proceedings of the 8th International Research Conference*, 2015.

[50] Stadt Kaiserslautern. Kaiserslautern (Website). `https://www.kaiserslautern.de/start/index.html.de`. [Online; accessed 2021-12-01].

[51] Jan Strehmann. Smart-City-Atlas: Die kommunale digitale Transformation in Deutschland. 2019.

[52] Guido Van Rossum. *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[53] Martin Verlage, Lara Kahl, and Patrick Torakai. Unser Lautern – Herzlich Digital – Deine Vision Unsere Zukunft – Leitstrategie für die digitale Transformation der Stadt. `https://www.kaiserslautern.de/mb/themen/projekte/kl_digital/pdf/20210115_strategie_herzlich_digital_einzelseiten_anschnitt.pdf`, 2020. [Online; accessed 2021-12-01].

[54] John Wonderlich. Ten Principles for Opening up Government Information. *Sunlight Foundation*, 2010.

# Appendix A

# Inventory of the Open Data Portals of Germany and Kaiserslautern

**Table A.1:** Dataset inventory (per category) in the open data portals of Germany and Kaiserslautern (as of 2021-12-01). Note that a dataset might belong to several categories.

| **Portal** | **GovData** | **OpenData Portal Kaiserslautern** |
|---|---|---|
| Population and Society | 13594 | 23 |
| Education, Culture, and Sport | 5073 | 1 |
| Energy | 4576 | 0 |
| Health | 2036 | 2 |
| International Topics | 261 | 0 |
| Justice, Legal System, and Public Security | 4937 | 0 |
| Agriculture, Fisheries, Forestry and Food | 4977 | 0 |
| Government and Public Sector | 11435 | 2 |
| Regions and Cities | 6847 | 6 |
| Environment | 11662 | 0 |
| Traffic | 9874 | 3 |
| Economy and Finances | 9690 | 0 |
| Science and Technology | 1885 | 0 |
| **Total** | **53141** | **37** |

# Appendix B

# Overview of Click That 'Hood's Datasets

**Table B.1:** Excerpt from Click That 'Hood's playable maps and their sizes.

| Map | Category | Number of requested areas |
|---|---|---|
| World | Major Airports | 888 |
| Northrhine-Westphalia | Municipalities | 396 |
| USA | Amusement Parks | 242 |
| London | Tube Stations | 141 |
| Toronto | City Parts | 140 |
| Hamburg | City Parts | 103 |
| Berlin | City Parts | 95 |
| Germany | States | 16 |
| Bavaria | Counties | 81 |
| Dresden | City Parts | 85 |
| Unna | Counties | 69 |
| California | Counties | 58 |
| Ulm | City Parts | 55 |
| Africa | Countries | 50 |
| Dusseldorf | City Parts | 50 |
| USA | Contiguous States | 48 |
| Frankfurt | City Parts | 46 |
| Münster | City Parts | 45 |
| Baden-Württemberg | Counties | 44 |
| Augsburg | City Parts | 42 |
| Potsdam | City Parts | 34 |
| London | Districts | 33 |
| China | Provinces | 32 |
| Freiburg | City Parts | 28 |
| Germany | Capitals | 16 |
| **Kaiserslautern** | **Districts** | **10** |
| Australia | States/Territories | 8 |