Dashboard solutions for building analysis optimization

Data analysis of environmental conditions and smart controls performance of building data and comparison of existing dash-boarding possibilities for data analysis

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Bachelor's thesis

Electrical and Automation Engineering

Spring 2023

Anna Kuznetsova

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Electrical and Automation Engineering
Author Anna Kuznetsova

Abstract
Year 2023

Subject Dashboard solutions for building analysis optimization
Supervisors Juhani Henttonen

Dashboards are an effective way to analyze complex structured data, enabling users to quickly get insights and effect performed analysis by editing its parameters. This thesis focuses on the development of several interactive dashboards, as well as the analysis of data using these dashboards. The process of creating these dashboards involves benchmarking existing solutions, selecting several solutions for implementation, and designing a web app with comprehensive analysis and an intuitive user interface. Additionally, various analytical methods are employed to identify patterns and trends in the data.

Overall, this thesis provides an in-depth examination of the development of interactive dashboards and the analysis of data using them. The results demonstrate the advantages of using interactive dashboards as a data analysis tool and can be used to inform future research in this field.

Keywords Smart controls, Data Analysis, Python, Web Development, Dashboards.
Pages 40 pages and appendices 1 page

Content

1 Introduction 1

2 Methodology 1

3 Theory 3

3.1 Data Analysis Tools 3

3.1.1 Spreadsheets 3

3.1.2 Databases 4

3.1.3 Self-service Data Visualization 4

3.1.4 Programming Languages 4

3.1.5 Big Data Tools 5

3.1.6 Cloud 6

3.2 Dashboards in Data Science 6

3.3 User interface – Principles of interactive visualizations 8

3.4 Python in Data Science 10

3.5 Statistical Analysis 11

3.6 Web Frameworks 12

3.6.1 Backend Frameworks 13

3.6.2 Frontend Frameworks 13

3.6.3 Fullstack Frameworks 13

4 Benchmarking 14

4.1 Evaluation 15

4.1.1 Self-service BI 15

4.1.2 Python 16

4.1.3 Web Frameworks 17

4.2 Comparison 18

4.2.1 Self-service BI 18

4.2.2 Python 19

4.2.3 Web Frameworks 21

5 Implementation 22

5.1 PyWebIO 23

5.1.1 Code structure 23

5.1.2 Visualization and functionality 25
5.2 Plotly Dash 28
5.2.1 Code structure 28
5.2.2 Visualization and functionality 30
5.3 Django and Plotly 32
5.3.1 Code structure 33
5.3.2 Visualization and functionality 34
6 Analysis 36
6.1 Smart Controls Performance 36
6.2 Dashboards performance 37
7 Conclusion 38
References 39

Figures, tables, and equations

Figure 1. Dashboard example (Calzon, 2021) 7

Figure 2. Top 10 Data Science Programming Language (Hayes, 2019) 11

Figure 3. Web frameworks graph (Ryabtsev, 2023) 12

Figure 4. Data Analysis Tools Comparison 14

Figure 5. SSBI solutions comparison 18

Figure 6. Python solutions comparison 20

Figure 7. Web Frameworks comparison (Khatri, 2023) 21

Figure 8. Source files structure 22

Figure 9. PyWebIO Project tree 23

Figure 10. Basic PyWebIO app 24

Figure 11. Basic PywebIO callback 25

Figure 12. PyWebIO Landing page 26

Figure 13. PyWebIO landing page after submitting form 27

Figure 14. PyWebIO Smart controls tab 28

Figure 15. Plotly Dash Project tree 29

Figure 16. Plotly dash main script 30

Figure 17. Plotly Dash page script 30

Figure 18. Plotly Dash home page 31

Figure 19. Plotly dash smart controls page 32

Figure 20. Django and Plotly Project tree 33

Figure 21. Django and Plotly home page 35

Figure 22. Django and Plotly graph 35

Terminology

Virtual Environment – isolated environment or simply a directory, where some executable files and scripts are installed. The virtual environment is used to isolate project files used for packages and external modules not to interfere with the operation of the system on which the application will run.

OOP – object-oriented programming, an approach in which the program is considered as a set of objects interacting with each other.

Data modeling (In software engineering) – the process of developing a data model for an information system.

pip - is a package management system that is used to install and manage software packages written in Python.

CSV (.csv) – comma-separated values – a type of text file which uses commas to separate values.

VS Code (Visual Studio Code) – a 'lightweight' source code editor for different programming languages and cross-platform development of web and cloud applications. The editor was developed by Microsoft and is compatible with Windows, Linux, and macOS.

JSON (.json) – JavaScript Object Notation – is a text-based data exchange format based on JavaScript, which is readable by both humans and machines.

HTML – HyperText Markup Language – a hypertext markup language used to view websites via browsers. Web browsers interpret HTML files they received via HTTP/HTTPS protocols from the server or got from the local disk into an interface and display it.

Cloud is an IT infrastructure, the term usually used to describe a way of big data storage on servers in data centers. Cloud computing is a delivery of IT resources.

Git – open-source version control system, originally developed in 2005 by Linus Torvalds, creator of the Linux operating system kernel. Many software development projects, both open-source and commercial, use Git for version management.

Web – a system for accessing related documents on different computers connected to the Internet.

UI – user interface, a tool for user interaction with a website, platform, or app.

UX – user experience, a process of recreating the user experience of interaction with a website to find flaws and improve UI.

Big Data – a term that refers to both structured and unstructured large data arrays.

Appendices

Appendix 1. Links to Dashboards

1 Introduction

This data analysis tools research was made part of the HAMK Smart Research Unit project in order to produce a comprehensive understanding about smart control features and performance installed for a heating system of the buildings provided by Talotohtori 2.0.

The aim of this thesis is to find tools which are capable of developing comprehensive dashboards, with possibilities to create advanced charts and a clearly defined structure of layout without a great deal of coding knowledge. In addition to being able to display data in a variety of plotting formats, such as lines, histograms, and different distributions, they must also have an interface that is easy to use and be able to display data quickly and efficiently. The potential for hosting is another crucial factor.

Stephen Few (2022) said: “Data visualization is just a tool. We could build houses before we had hammers and saws, the tools just let us do it better”.

2 Methodology

This thesis is separated into five parts: Theory, Benchmarking, Implementation, Analysis and Conclusions. Each part is needed to achieve two main goals: the analysis of the provided buildings data and the identification of convenient solution for creating informative and efficient reports and dashboards.

The theory chapter provides a background knowledge and terminology necessary for understanding the subsequent chapters. The theory chapter defines analytics tools, platforms and dashboards and how they are created.

The benchmarking chapter includes assessment, evaluation and comparison of three identified and selected categories of dashboarding tools. A comparison table of the existing solutions for each category was made. Every table includes unique comparative metrics that are significant for each table's category. Further, specific solutions are chosen, and the justification is given.

The section on implementation provides a summary of the three options that were chosen and put into practice. Its code structure is briefly described and provided. Then, for better comprehension, a description of functionality and visualization structure is provided with screenshots.

The analysis section is where generated solutions are examined, with the effectiveness and overall performance being assessed. Visualizations will be examined to look for patterns and trends, in order to study the performance of smart controls, the surroundings and the outcomes provided by each solution. This may entail investigating outliers, correlating various variables, and discovering any significant insights.

Additionally, the selected dashboards will be analyzed. When analyzing dashboards, there are several key factors to consider. First, consider how easy the dashboard is to use and interpret. Is the design intuitive? Are the visualizations clear and easy to interpret? Are there any navigation issues?

Second important factor is entry threshold required to realize dashboard potential fully. How much knowledge and skills required to develop a dashboard.

Another important factor to consider when analyzing dashboards is the ease of data handling and hosting possibility. Was it easy to work with data provided? How dashboard can be acessed by other people?

Finally, consider how the dashboard is being used. Is the dashboard being used to its fullest potential? Are users taking advantage of all the features? Are users finding the dashboard to be helpful in achieving their goals?

By taking all of these factors into account, it is possible to gain an understanding of how effective a dashboard is. An analysis of developed dashboards will help ensure they are providing the most value and provide insight into any areas of improvement.

The analysis is evaluated and summarized at conclusion chapter, and it is determined whether the performance of smart controls was efficient and whether the final results

allowed for any conclusions to be drawn or statements to be made regarding the conditions of the building environment.

Additionally, it is determined whether implemented solutions met expectations, whether created solutions can be deemed feasible and efficient, whether best solution can be identified, whether they require improvement, and whether the thesis' objectives were met.

The most important research questions thesis should answer are:

- What options does market now provide for data analysis?
- Is there a universal solution for creating interactive analysis dashboards which offers relatively low entry threshold and ease of access to the application without limiting functionality?
- What conclusions about smart controls performance and environment conditions of the buildings can be done?

3 Theory

This part of the thesis will overview all implemented knowledge and theoretical justifications which led to the implemented solutions.

3.1 Data Analysis Tools

Data analysis tools can be separated into several categories: Spreadsheets, Databases, Self-service Data Visualization, Programming Languages, Big Data Tools, and Cloud solutions. (Ng, 2018)

3.1.1 Spreadsheets

The spreadsheet is a general term for programs and platforms which allow you to operate with data in form of a two-dimensional array. Visually they imitate a paper table and offer users to perform calculations, reorganization of data, formatting of visual representations,

and plot. Some also organize data into ‘sheets’ (third dimension). The most common Spreadsheet platforms are Excel, Numbers and Google Sheets (Langmann, 2023).

3.1.2 Databases

Databases –structured data that is stored by defined schema electronically in a computer system. Manipulating and operating with data is performed by the principles and algorithms of data modeling tools.

3.1.3 Self-service Data Visualization

Self-service Data Visualization or Self-service Business Intelligence (SSBI) is a concept where a business analysis platform is oriented on providing business users with convenient tools for independent and effective data analysis. The functionality, such as filtering, grouping and plotting tools, should allow a user with no programming skills to create their own customized, depending on their needs and field, dashboards, and reports. The most popular Business Intelligence Tools are Power BI and Tableau (SelectHub, 2023).

3.1.4 Programming Languages

Programming languages are widely used in data science. They allow users with knowledge of the particular language syntax to manipulate, structure, and visualize data according to their needs. The most widely used languages in data analysis are Python, R, SQL, Java, Scala, Julia, and Matlab (Kumar, 2018).

Python – is one of the most popular languages among data analysts. Offers a wide range of modules and libraries for data handling, data operation, analytics, statistics, visualization, and even machine learning. Great examples are pandas, numpy, matplotlib, and TensorFlow.

R - language and environment for statistical computing and graphics. Also, widespread because it has packages for any quantitive and statistical application.

SQL – structured query language, used to define, manage and query relational databases. SQL is easy to read, used in a wide range of applications, and can be integrated with other languages.

Java – a standard language of general purposes with a powerful ability to integrate data science and analytics techniques. Runs on the Java Virtual Machine (JVM).

Scala – multiparadigm language runs on JVM and allows users to use object-oriented and functional approaches.

Julia – high-level programming language, which was originally designed for mathematical computing, it can also be used for general-purpose programming. Julia offers simplicity, dynamic typing, speed and scripting capabilities.

Matlab - programming language and platform for iterative analysis, process design, programming, and numeric computing. Used in academia and data processing industries where quantitive applications that have complex mathematical requirements are needed.

(Kumar, 2018)

3.1.5 Big Data Tools

Big Data Tools are software for efficient big data processing which helps to achieve a competitive advantage in the market. With big data analytics software, you may keep track of client requests, market trends, and other important data. The most popular tools are Hadoop and Spark (Hillier, 2022).

Apache Hadoop is a big data storage system that is open source, and free and comes with a variety of tools, libraries, frameworks, and development packages. This foundational big data storage and processing technology is a top-level project of the Apache Software Foundation. Hadoop consists of: HDFC (distributed file system), MapReduce (distributed computing model by Google), YARN (clusters managing technology), and libraries.

Apache Spark is a big data platform that uses distributed in-memory computing to accelerate processing. It uses various forms of computations, such as interactive queries and stream processing, and is based on Hadoop.

3.1.6 Cloud

Cloud computing is a concept according to which all apps and the data they require for work are stored on a remote server on the Internet, but programs are launched and output the results of their work to a window of a regular web browser on a local computer.

Cloud solutions are a distributed data processing technology that makes use of cloud computing by providing computer resources and capacities to the user as a service.

There are three different service models of cloud solutions:

- Infrastructure-as-a-service (IaaS), the customer uses the computing resources of the provider.
- Platform-as-a-service (PaaS), the supplier gives customers access to a software platform.
- Software-as-a-service (SaaS), the customer uses a ready-made application from a supplier.

The most popular solutions are:

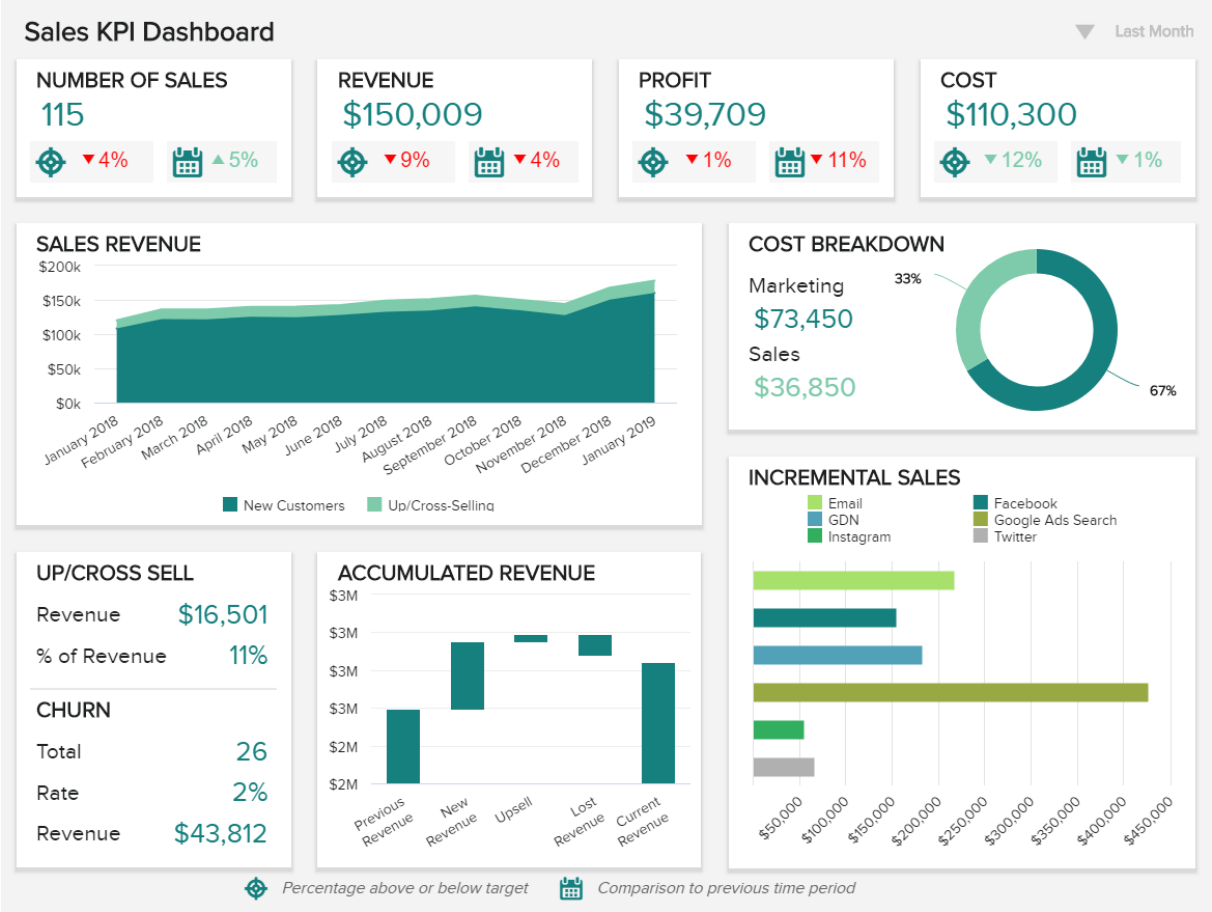
- AWS by Amazon
- Azure by Microsoft
- Google Cloud Platform

3.2 Dashboards in Data Science

A dashboard Is an analytical panel, which is a synthesis of real-time data collection, mathematical analytics methods, and an optimal graphical representation of analysis results.

Important elements of any dashboard are key performance indicators, trends, dependencies, etc. Metrics should be displayed in an understandable compact form with possibilities of interactive change of different parameters used in analysis or graphics. In addition to the visualization of data, one of the goals achieved with the help of dashboards is related to the comparison of one or another indicator over time, or assessment relative to other indicators.

Figure 1. Dashboard example (Calzon, 2021)



The main difference between a dashboard and a report is that a report captures certain numbers and data over a specified time or period, e.g. monthly or annually. Meanwhile, dashboards, due to their modularity and widget variety, can monitor important indicators in real-time and dynamically change the widget's priority to update data hierarchy and visualization format depending on current user priorities. Moreover, it allows for the comparison of any values or data blocks over a selected time.

Three types of dashboards can be indicated depending on their initial purpose: process management (to identify weak points and find solutions), process analysis (to identify the efficiency of implemented solutions), and process monitoring (to track the current state in real time).

The main methods used when creating a dashboard or analytic report are:

1. Grouping – combining similar data by some of the common features (name, weekday, date, etc.)
2. Sorting – ordering data according to given attributes (alphabetic order, ascending order, descending order, etc.)
3. Filtering – exclusion of data by given attribute (more than zero, only with word 'temperature' in the column name, only integer type, etc.)
4. Aggregation – getting facts and numbers from the initial database (minimum, maximum, average, etc.)
5. Calculating column – obtaining new column with data using methods of working with datatypes or mathematical functions (getting weekday name from date, calculating outside and inside temperature difference, converting foot to centimeters, etc.)
6. Widgets – actual way to visualize the above concepts (tables, dropdown menus, date pickers, graphs, etc.)

The dashboard is the interface between the analytics engine and whoever acts as the user analyst. Thus, all the principles of building interfaces, methods of improving UX, and increasing Usability apply to the dashboard.

3.3 User interface – Principles of interactive visualizations

Data visualization is a graphical representation of information and analytics: graphs, charts, maps, and dashboards. The numbers in the data do not clearly show the relationship between processes, dependencies of indicators, or periods of growth. The visual format presents information, puts everything in order, helps to understand it faster, and

emphasizes trends. But that's only assuming that the given tools and methods were used correctly and effectively.

Eight core principles of effective interactive visualization tools were formulated by Stephen Few (for both tool developers and dashboard creators):

- Simplify – each indicator is understood unambiguously and shown clearly and simply.
- Compare – It should be obvious which values are compared; they should be shown next to each other.
- Attend – important data is shown in the first place, easily available.
- Explore – find new ways.
- View Diversely – view the same data in several different ways, giving different perceptions.
- Ask why – give an understanding of why results are like that and show reasons.
- Be skeptical – ask more questions to inspect the subject deeper.
- Respond – give answers in your visualizations.

(Few, 2022)

A few more principles can be added:

- Avoid visual interference – an overload of visual elements can distract the user from important information or distort the result of data analysis.
- Supplement visualization with text – visualizations should be perceivable and understandable on their own, but it is a good practice to complement them with text.
- Take into account features of human perception – people combine some of the visual parameters of an object into an 'image'.

To summarize, three main rules can be identified:

1. Logic: firstly, the purpose of the visual element should be formulated to determine what will it present; secondly, the task should be identified to know how to achieve the desired view; next, data are logically systematized and visualized taking into

account the ease of perception. Data should be arranged in a logical order depending on the purpose. When comparing, charts of the same type should be used.

2. Structure: design decisions should be consistent with the purpose of creation and the characteristics of the data and be concise - this improves perception, helps the decision, and does not distract.
3. Color: stick to a single-color scheme. Take into account the 'common colors' of target users.

3.4 Python in Data Science

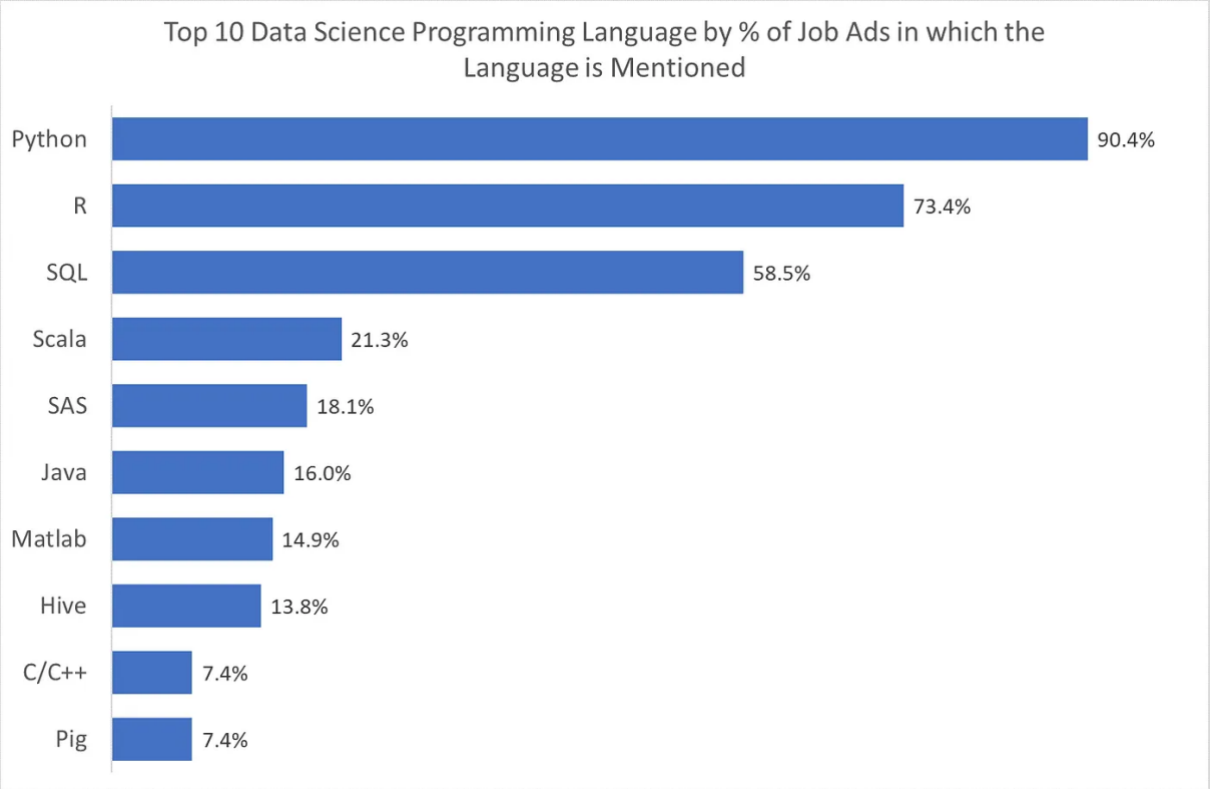
Python is a high-level programming language that is efficient, easy, and versatile to use. It is widely used in web and application software development, as well as in machine learning and big data processing. Due to its simple and intuitive syntax, it is one of the most common languages for learning to program.

Math is the biggest part of data science; thus, data scientists need a simple language with wide functionality. At this point, python has many strengths:

- Hight productivity – python has a simple syntax. This allows you to write code faster than in other programming languages (such as Java or C). Moreover, the Python code is readable and easy to interpret.
- Low entry threshold for learning – takes a minimum of time and money to learn.
- Integrated features for code optimization – built-in interpreter allow to dynamically determine the optimization solutions during code execution.
- Dynamic language development – python developers are open to the community and take into account users' ideas. With each new version, the performance of the language increases, and the syntax improves.

According to research of Genevieve, Hayes Python is by far the most relevant programming language for data science. It is mentioned in 90.4% of job offers for Data scientist vacancies (Hayes, 2019).

Figure 2. Top 10 Data Science Programming Language (Hayes, 2019)



Python offers primary data analysis in pandas, calculations in NumPy/SciPy, machine learning in sklearn, and neural networks in TensorFlow or PyTorch. All libraries are easily installed with pip.

3.5 Statistical Analysis

The science of gathering data and identifying patterns and trends is known as statistical analysis.

Six types of statistical analysis can be defined:

- Descriptive Analysis
- Inferential Analysis
- Predictive Analysis
- Prescriptive Analysis

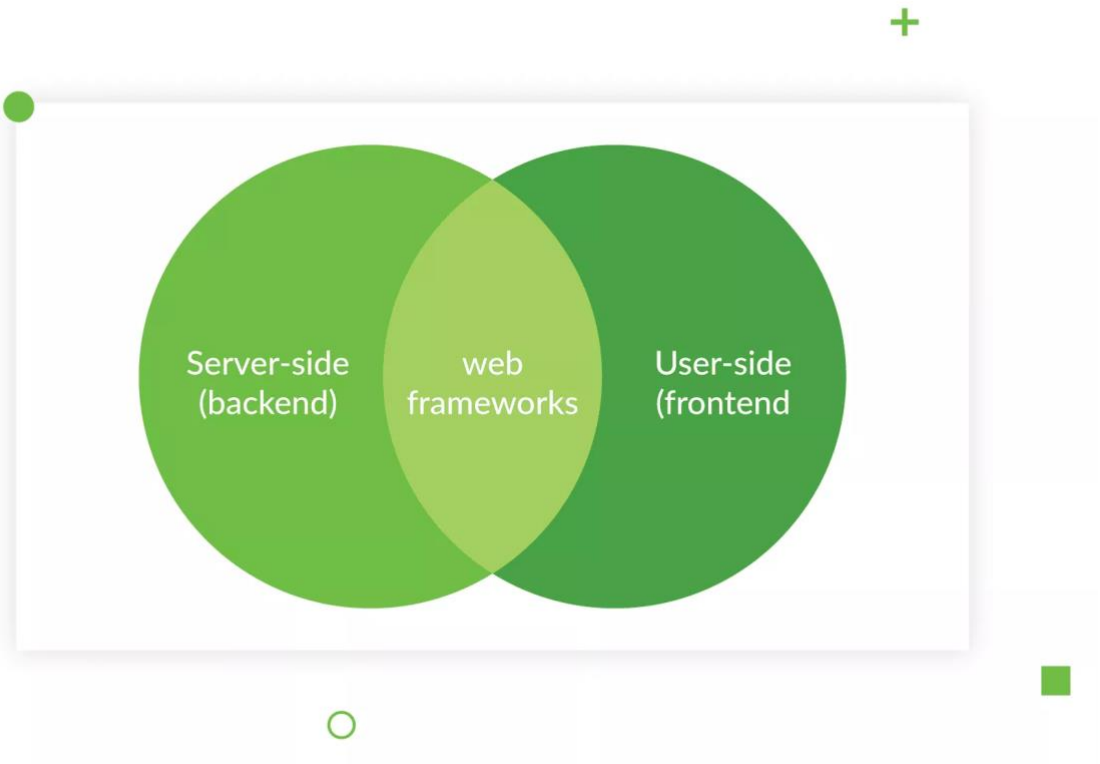
- Exploratory Analysis
- Causal Analysis (Simplilearn, 2023)

It is difficult to define the precise boundary between statistics and data analytics, because statistical techniques are the foundation of data analytics. However, for data analysis, mathematical precision is not as crucial.

3.6 Web Frameworks

A framework for creating web applications is called a web framework. It supplies the essential set of development tools, establishes the rules, and defines the structure. Two factors can be used to categorize web application frameworks: the jobs they handle and their size.

Figure 3. Web frameworks graph (Ryabtsev, 2023)



3.6.1 Backend Frameworks

These are server-side frameworks for web development. In essence, they are in charge of crucial parts of the application that would not function properly without them. The rules and architecture of server frameworks make it impossible to develop a web application with a sophisticated user interface. Although their functionality is constrained, you can still make simple websites and forms. Additionally, they can provide output data and manage security. The most well-known frameworks and the languages they work with are listed below:

- Python – Django
- JavaScript – Express.js
- Ruby on Rails

3.6.2 Frontend Frameworks

The appearance of a web application is handled by front-end frameworks. They are unrelated to the logic of work, unlike server ones. These frameworks function in browsers. They allow to development and improve user interfaces and create various animations, single-page apps, and designs. These are a few of them:

- Angular
- React.js
- Vue.js

3.6.3 Fullstack Frameworks

The framework falls under the complete stack category if it resolves issues on both the server and client sides. Meteor is a good illustration. JavaScript is used by both its client and server sides; thus, you may write and apply the same code to them. "Real-time mode" is the following functionality.

4 Benchmarking

This chapter is dedicated to selecting tools for further implementation.

After careful consideration, two tools were selected for benchmarking: Self-service business intelligence tools and programming languages. Further, it was decided to present the category of programming languages in the form of two groups: a pure language with its modules and libraries and web frameworks in this language. For this purpose, python was selected as the most popular and easy to acquire. Thus, defined categories are Self-service tools (BA), Python (with modules and libraries), and Web Frameworks (on Python). Categories are summarized in the table below (**Error! Reference source not found.**).

Figure 4. Data Analysis Tools Comparison

	Hosting & accessing	Cost	Entry threshold	Responsiveness of final dashboard	Data handling
Self-service BI	Cloud share, Web Server	Not free	Not required	Weak	Built-in
Python	Git, Web	Free	Knowledge of Python	Strong	External modules
Web Frameworks	Git, Web	Free	Knowledge of Python and HTML&CSS	Strong	External modules

4.1 Evaluation

In this subchapter, selected categories will be viewed and evaluated, and their advantages and disadvantages will be considered.

4.1.1 Self-service BI

As was previously stated, ‘self-service BI’ is a term used to characterize platforms and apps for creating an interactive analysis of data without the programming skills required. The main advantages of this data analysis tool can be easily defined:

- Effective use and distribution of IT and business analysis resources. Self-service allows corporate users to perform their own ad hoc analysis. The majority of queries, visualizations, dashboards, and reports may be created by the organization's business users without the need to involve the IT department of a company. As a result, they may concentrate on high-priority jobs that call for more technical expertise, such as managing datasets for business customers and building sophisticated queries.

- Rapid and responsive decision-making. By distributing the analytical effort to business users rather than a limited group of business analysts, self-service capabilities assist in identifying bottlenecks in business intelligence applications. As a result, company operations move more quickly because users are better able to assess data, make judgments, and take action.

- Data-driven company organization and competitive advantage. Self-service solutions stimulate the development of a data management culture by making it available for everyone. Utilization of data leads to faster decision-making, thus, increasing competitive advantage.

Without a doubt, all listed advantages make SSBI a very attractive option, but one cannot ignore the shortcomings that come with this solution:

- Disorganization. Companies risk having a chaotic set of metrics that differ throughout departments if you encourage your business users to become committed data engineers.

- Security issues. Since Self Service is typically used to create custom analytics applications, the IT service must set up access so that users can only access the information they need. This includes allocating server space for application data and configuring the ability to delete such applications after a set period so that they do not strain server capacity.

- Need for standardization. As business analysts are responsible for SSBI tools, they should also manage and standardize the reports that managers will produce to guarantee that the outcomes are accurate and significant throughout the company.

4.1.2 Python

Python works great on all stages of data analysis; various libraries help with this. Of course, there are no perfectly suitable languages for data analysis, some languages better handle visualization, and some machine learning. Let's overview the advantages of Python for data analysis:

- Wide community. There are many communities, blogs, and websites that help find solutions to various problems and tasks, as well as learn new things.

- Simple syntax and readability. Python is highly human-readable which makes it easy to learn. It also requires far fewer lines of code.

- Variety of libraries. A huge number of libraries can be used at any stage of data analysis. There are libraries for reading, manipulating, and handling data, for conducting complicated mathematical and statistical calculations, and for machine learning and creating responsible visualizations. Moreover, most of the libraries are free.

As for the downsides, one of the main disadvantages of Python in data analysis are its speed (JuliaLang.org, 2023) and dynamic typing. Python was not developed specifically for data analysis; it is a general-purpose language. Dynamic typing makes development considerably simpler, but it makes finding mistakes in data associated with different types difficult.

4.1.3 Web Frameworks

Although Python Web Frameworks undoubtedly have all advantages of the language itself, there are a few more that may be mentioned when discussing frameworks:

- Embeddability and integration with other languages. Python is ideally suited for dynamic semantics and rapid prototyping and can be easily integrated into a wide range of applications, even those that use different programming languages.

- Simplified implementation of OOP. The work of OOP is substantially simplified in Python, which reduces the cost and duration of development.

Despite its apparent advantages, Python, in terms of web development, also has certain drawbacks that should be mentioned:

- Limited speed. Since Python is a high-level language, it is relatively slower than C/C++ or Java, which do not need to spend time translating program text. This disadvantage does not interfere with the operation of simple programs or scripts, including those for data analytics, however, it can insignificantly affect the operation of large web projects.

- Lack of multithreading. It might be not as adaptable or practical as other languages and when running the code in parallel, might lead to some challenges.

Python is nevertheless regarded as one of the top programming languages for new businesses. Python may be used for developing and launching minimal viable products since it is versatile, quickly scalable, doesn't require a huge team, and is easy to learn.

4.2 Comparison

After considering and evaluating the selected categories, existing solutions from these categories can be benchmarked.

4.2.1 Self-service BI

In the Self-service BI category, there are many great solutions. For comparison were selected: Power BI, Tableau, and Oracle Analytics Cloud were. Criteria, pros, and cons were summarized in the table below.

Figure 5. SSBI solutions comparison

	Power BI	Tableau	Oracle
Price per month per user, $	10	15	16
OS	Windows	Windows	Windows, MacOS
Web	Server	Server	Additional product required
Community and Documentation	Documentation and forum on the Microsoft website	Tableau forum and functions documentation inside the application	Documentation on the website, workshops

It was decided not to take Self-service BI tools for modeling because this is not a free source software.

4.2.2 Python

Jupyter Notebook – notepad program (or can be used as a development environment) for writing, transferring, and running code. Because the code can be divided into smaller fragments and executed in any sequence, it differs from the conventional development environments. It exists as a web service that is available over the Internet and allows you to transfer code to other developers.

Plotly Dash – a Python open-source analytical platform for building web browser data visualizations, interfaces, dashboards, and forms. Plotly Dash offers comprehensive documentation on how to create low-code apps, scale them and deploy them. Plotly Dash is considered to be one of the leading low-code platforms.

Streamlit – Python low-code web framework for creating web apps. Has a much simpler syntax than Plotly Dash.

PyWebIO – a relatively new low-code web framework designed to create simple web applications without HTML or JavaScript knowledge.

Tkinter – is a cross-platform package of Python modules for building GUI desktop applications.

PyQT – is a cross-platform Python library for building GUI applications using the Qt toolkit.

The advantages and shortcomings of Python tools that can be used for dashboard development were summarized in the table below. For UI/UX parameters 1 to 5 scale was used, where 1 is insufficient, 3 – is sufficient, and 5 – is more than satisfactory.

Figure 6. Python solutions comparison

	UI/UX	Threshold	Architecture	Deploy method	Community & Documentation
Jupyter Notebook	UI/UX	Python	Easy to structure code	Local, share, web with external tools	Documentation, community meetings
Plotly Dash	5/5	Python, HTML&CSS	Not easy	Deploy on platform	Documentation, Demos, blogs, webinars
PyWebIO	4/5	Python	Can be messy but structured	Side platforms	Documentation, tutorials, demos
Streamlight	4/5	Python	Easy	Deploy on platform	Documentation, cheat sheet, blog, demos
Tkinter	3/5	Python	Mild, structured	Desktop	Documentation (in form of tutorials), book
PyQT	4/5	Python	Mild, structured	Desktop	Documentation, Forum, Support service

It was decided to put two of these options into practice. Plotly Dash was chosen for its extensive visualization capabilities, popularity, and compatibility with Bootstrap, and PyWebIO was chosen for its simple modular structure and thorough documentation.

Streamlit was not selected because it has fewer customization options than Plotly Dash. It also loses in terms of speed and performance (Hwang, 2020). Both Streamlit and PywebIO are low-code solutions that don't require any front-end frameworks background. PywebIO was chosen because it is more lightweight and has more thorough documentation.

4.2.3 Web Frameworks

Django – is a free framework for developing fast and secure web applications and websites in Python.

Flask – is a microframework for creating a simple and fast project in the Python programming language with the ability to scale your apps further.

Figure 7. Web Frameworks comparison (Khatri, 2023)

	Django	Flask
Type	Full stack	Micro-framework
Flexibility	Low	High
Popularity	7.89% (Developer Survey: Stackoverflow, 2022)	4.32% (Developer Survey: Stackoverflow, 2022)

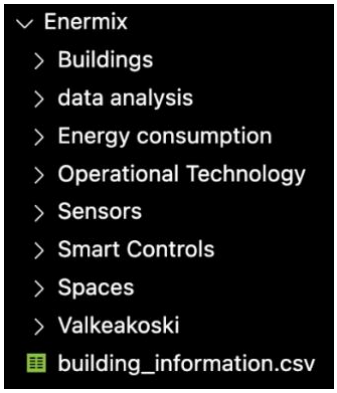
Bootstrapping tool	Yes	No
Multi-Page Apps	Yes	No
Dynamic HTML	Yes	No

Even though Django requires more skills, it was selected over Flask due to its compatibility with Bootstrap components and the possibility to create Multi-Page Apps.

5 Implementation

This chapter describes implemented solutions and their functionality comprehensively to compare them further. Every solution had same starting files with the same structure. Data folder called 'Enermix' has 8 folders and building information csv file which has basic information on every building (See Figure 8). Energy consumption, data analysis, operational technology, sensors, smart controls and Valkeakoski are folders with csv file for each building. Buildings folder consists of folders named after building id, in each folder there are json files with information on devices and spaces they are located in. Spaces folder has json files with data on every space in the buildings.

Figure 8. Source files structure



Each solution runs on a local host and follows the identical project-running instructions. When the main file has finished running, the terminal will produce a local host link that may be put into the browser.

Solutions, their source codes and documentation can be found on GitHub.

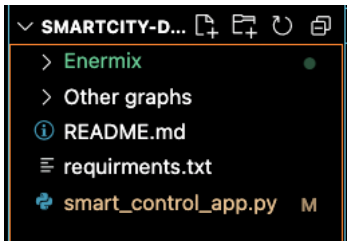
5.1 PyWebIO

Except PyWebIO library, some other libraries were used to create this solution, e.g. pandas, bokeh, pathlib, etc.

5.1.1 Code structure

Project files are structured according to Figure 9. PyWebIO Project tree.

Figure 9. PyWebIO Project tree



Requirements file has list of all libraries and extensions needed in order to run app. PyWebIO has a simple project tree no additional files for visualization or assets are needed.

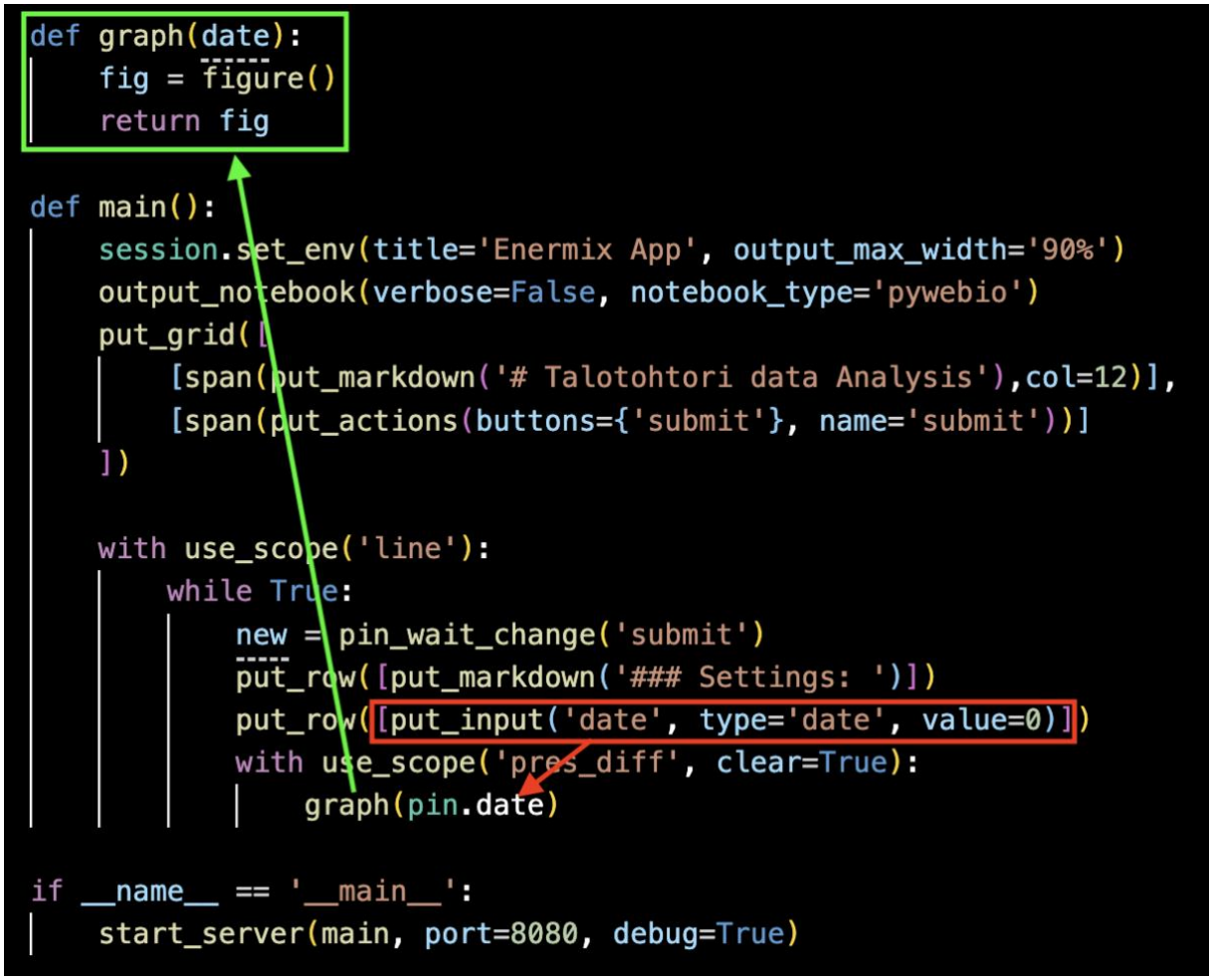
PyWebIO has a simple code structure. to build a simple app Two lines are required to run the server, and three lines of main function are required to build a simple web page (See Figure 10. Basic PyWebIO app).

Figure 10. Basic PyWebIO app

```
def graph(date):
    fig = figure()
    return fig

def main():
    session.set_env(title='Enermix App', output_max_width='90%')
    output_notebook(verbose=False, notebook_type='pywebio')
    put_grid([
        [span(put_markdown('# Talotohtori data Analysis'),col=12)],
        [span(put_actions(buttons={'submit'}, name='submit'))]
    ])

    with use_scope('line'):
        while True:
            new = pin_wait_change('submit')
            put_row([put_markdown('### Settings: ')])
            put_row([put_input('date', type='date', value=0)])
            with use_scope('pres_diff', clear=True):
                graph(pin.date)

if __name__ == '__main__':
    start_server(main, port=8080, debug=True)
```

Figure 11 shows callback logic from widgets to function, which then returns or updates visual elements to app layout.

Figure 11. Basic PywebIO callback

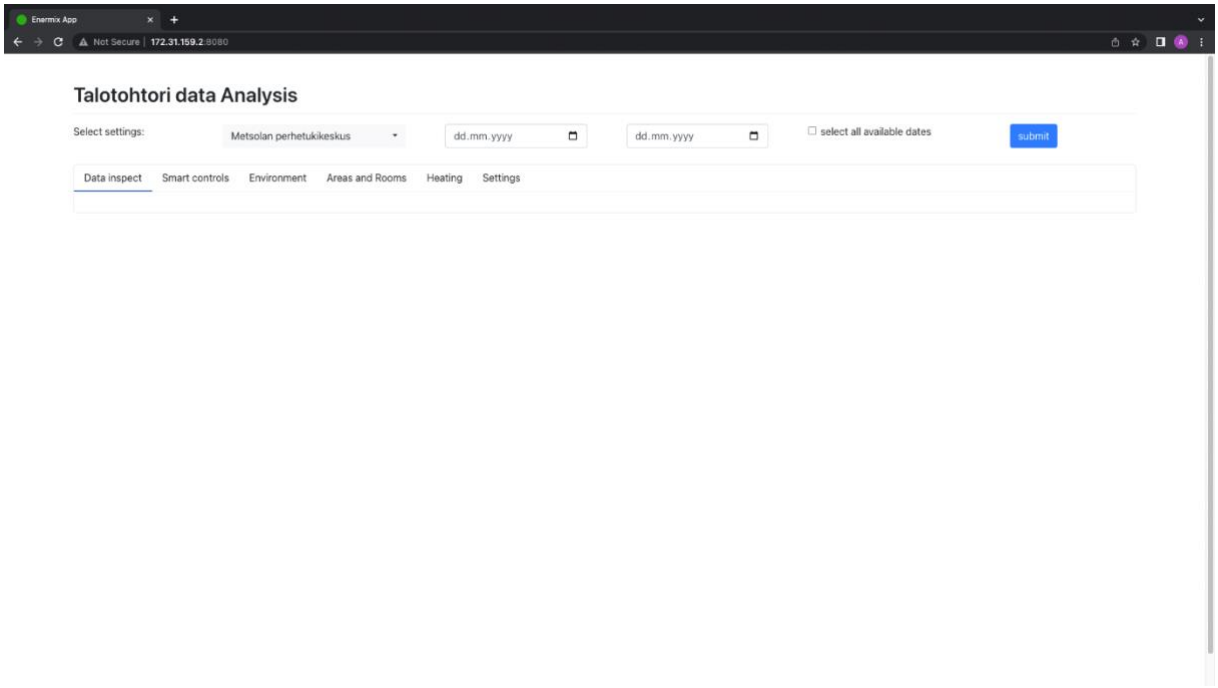


This callback mechanism is the foundation of the entire project. More components, features and functions were added. The outcomes for these are examined in the following chapter.

5.1.2 Visualization and functionality

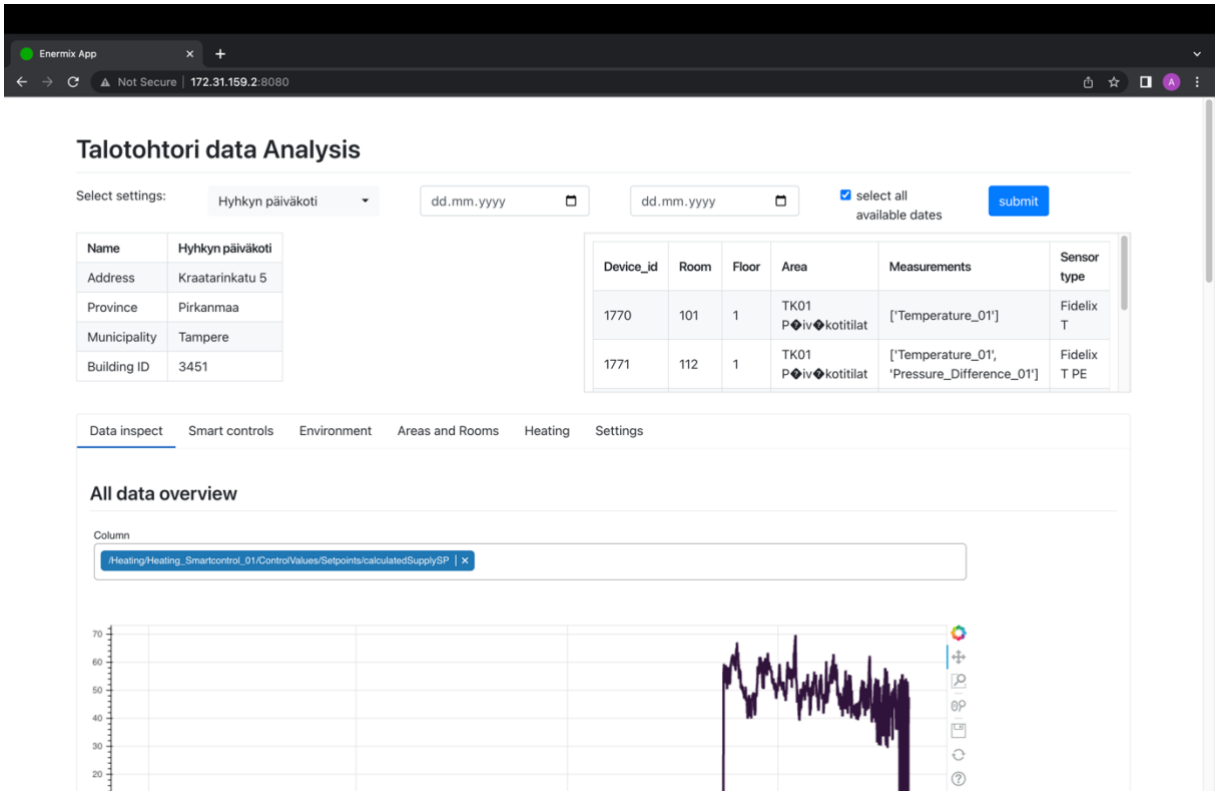
The landing page for this solution contains a title, a dropdown menu, two date pickers, a check box widget, a submit button and six tabs (See Figure 12. PyWebIO Landing page).

Figure 12. PyWebIO Landing page



The program changes the content of six tabs once the building and date range have been chosen and submitted. It also adds table with building information and table with devices installed in this building and their parameters. (See Figure 13. PyWebIO landing page after submitting form)

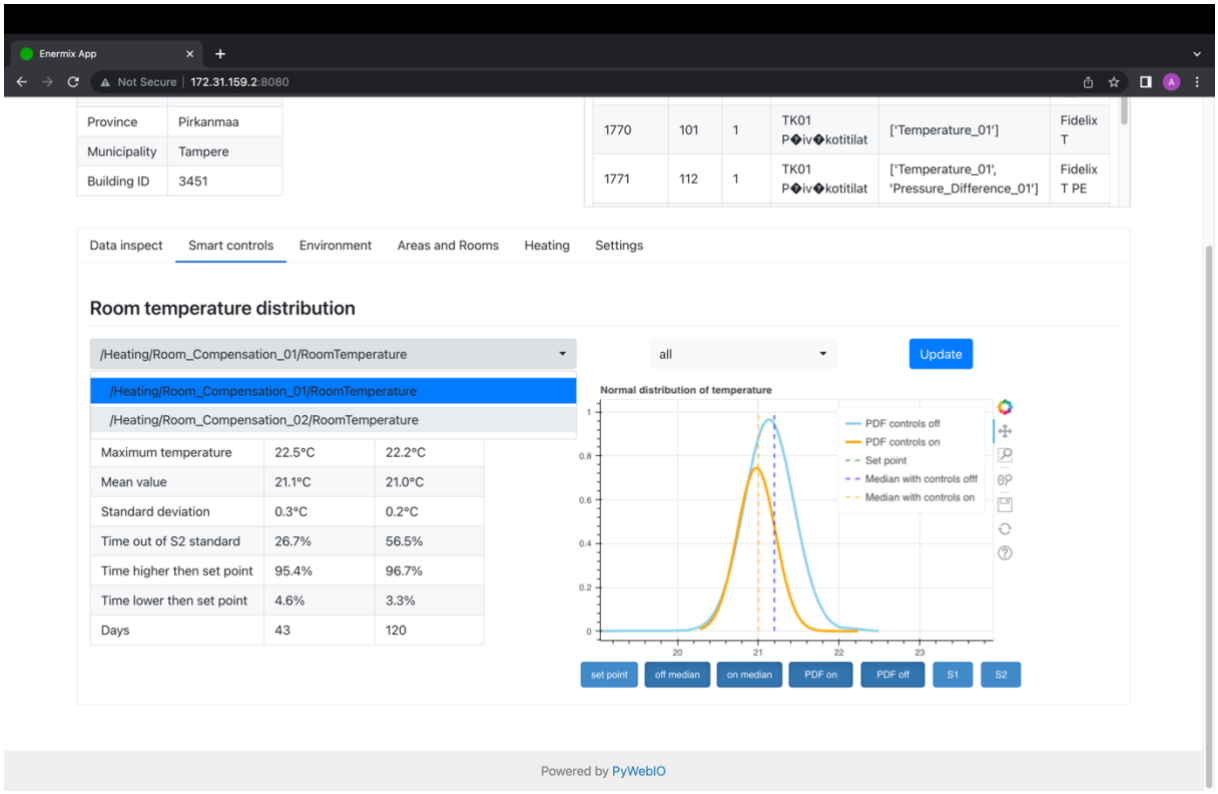
Figure 13. PyWebIO landing page after submitting form



In order to preview data and data quality, the Data Inspect tab contains five general-purpose graphs with all the data about the selected building across all folders presented as line plots. Graphs are created with Bokeh library and have ability to handle multi select function.

The smart controls tab has two drop-down menus for choosing the room and time frame (office-hours, weekends, weekdays, etc.), a submit button, a statistics data table, and a distribution graph. Distribution graph has additional widgets to add and remove some graph elements. (See Figure 14. PyWebIO Smart controls tab)

Figure 14. PyWebIO Smart controls tab



The app also has Environment tab with analysis of environmental condition, Areas and Rooms tab for reviewing conditions in particular rooms and areas, Heating tab where supply temperature is analyzed with respect to outdoor temperature and Settings tab where definition of office hours can be edited, and devices data can be updated.

5.2 Plotly Dash

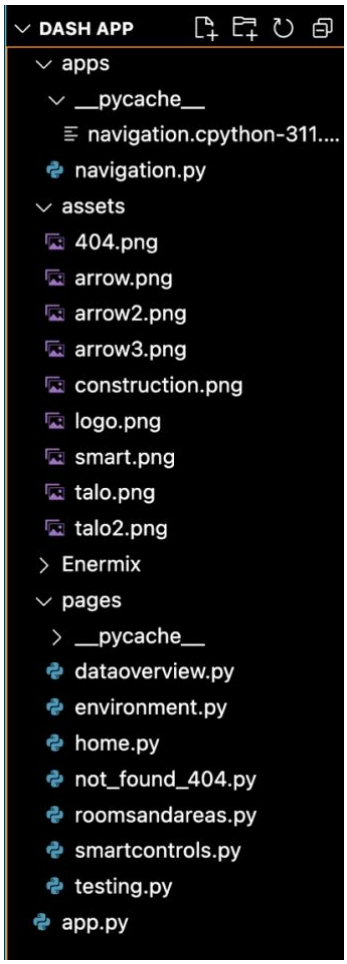
In this approach, charts are created using Plotly and Plotly Express, layout and widgets are created using Plotly Dash and Bootstrap components, and data handling and mathematical operations are performed using pandas and numpy lybraries.

5.2.1 Code structure

Project tree consists of four folders and main programm called ‘app.py’. The assets folder is where important images and logos are kept. Buildings data is in the Enermix folder. All the

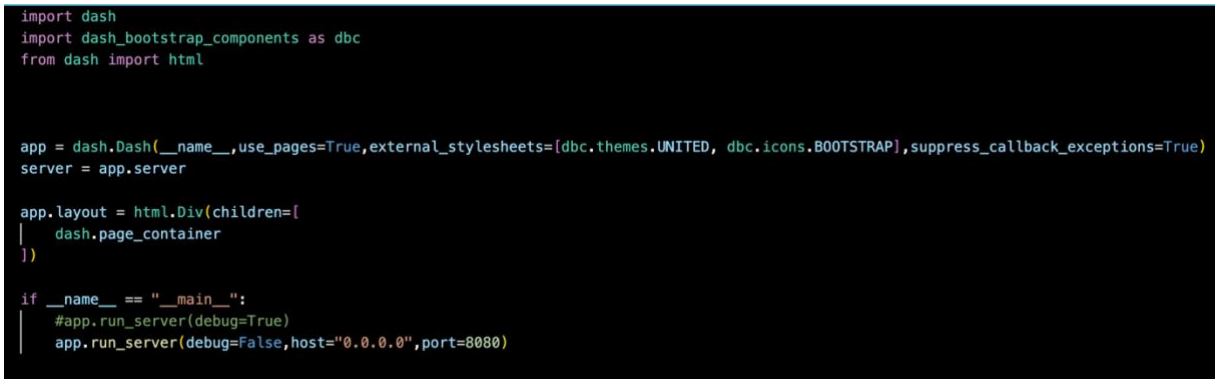
components and modules required for the layout are in the apps folder; in this case, it is the navbar element. Python files for each app page may be found in the Pages folder. (See Figure 15. Plotly Dash Project tree)

Figure 15. Plotly Dash Project tree



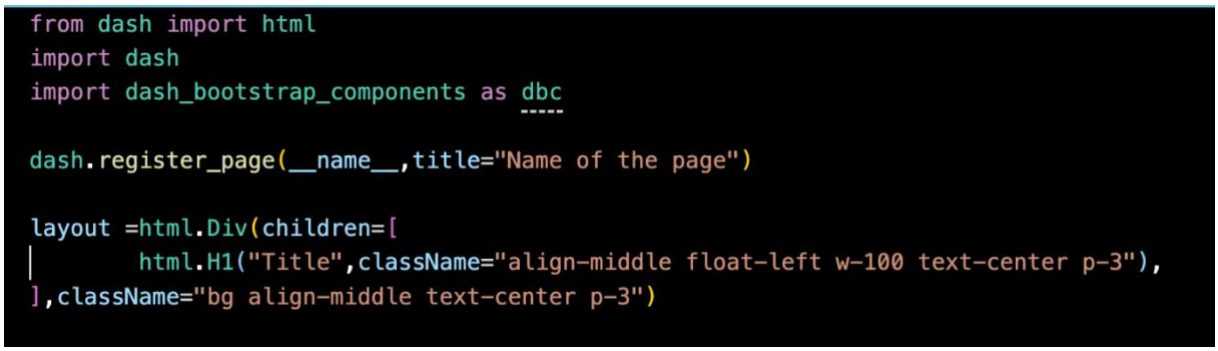
Six lines of code are required to construct a simple app (See Figure 16. Plotly dash main script). In order to add a page, a separate python file should be created in the page folder and page registered. Page registration takes one line of code (See Figure 17. Plotly Dash page script).

Figure 16. Plotly dash main script



```
import dash
import dash_bootstrap_components as dbc
from dash import html



app = dash.Dash(__name__,use_pages=True,external_stylesheets=[dbc.themes.UNITED, dbc.icons.BOOTSTRAP],suppress_callback_exceptions=True)
server = app.server

app.layout = html.Div(children=[
    dash.page_container
])

if __name__ == "__main__":
    #app.run_server(debug=True)
    app.run_server(debug=False,host="0.0.0.0",port=8080)
```

Figure 17. Plotly Dash page script

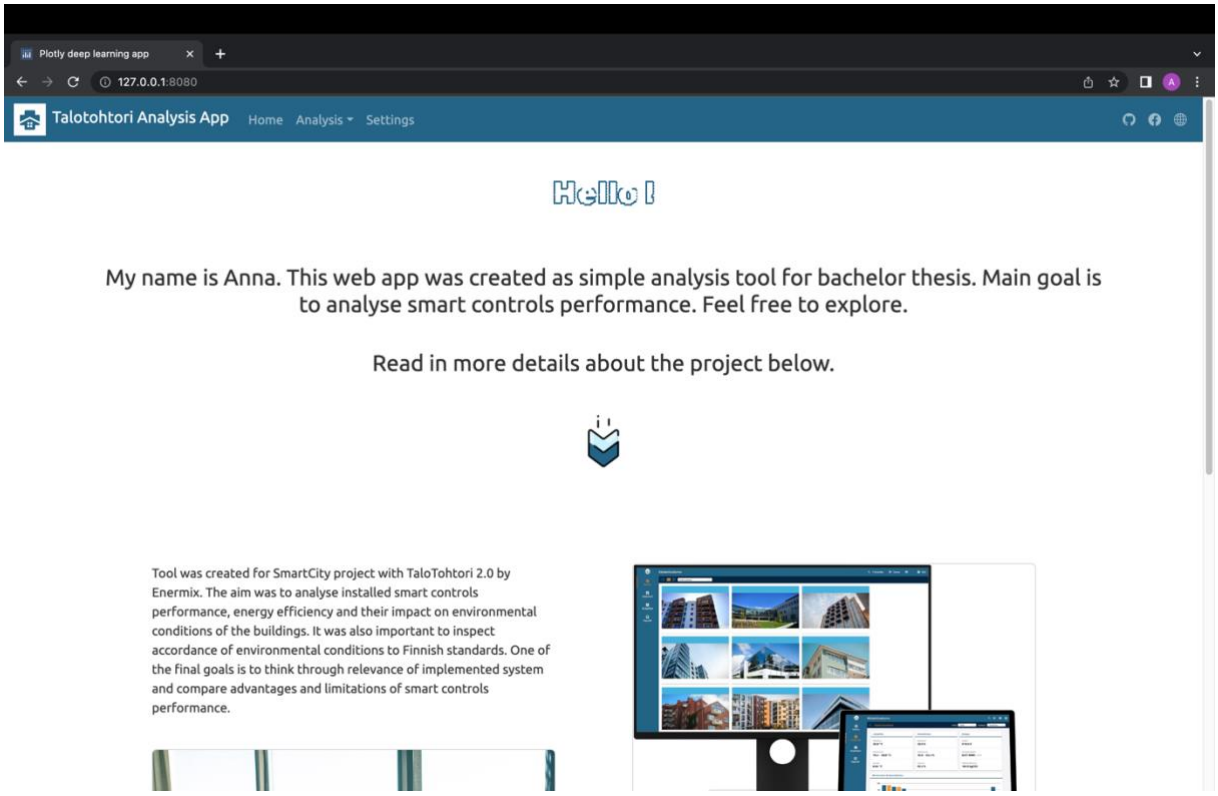


```
from dash import html
import dash
import dash_bootstrap_components as dbc

dash.register_page(__name__,title="Name of the page")

layout =html.Div(children=[
        html.H1("Title",className="align-middle float-left w-100 text-center p-3"),
],className="bg align-middle text-center p-3")
```

5.2.2 Visualization and functionality

This solution's visualisation consists of a navigation bar and seven pages. The navigation bar features a logo, title, two links to pages, a dropdown menu with access to further pages, and three icons with links to a website, a GitHub account, and a Facebook page. Home and Settings pages can be seen on navigation bar. A dropdown menu called Analysis can be used to access the Data Overview, Smart Controls, Rooms and Areas, Environment, and Testing pages. (See Figure 18. Plotly Dash home page)

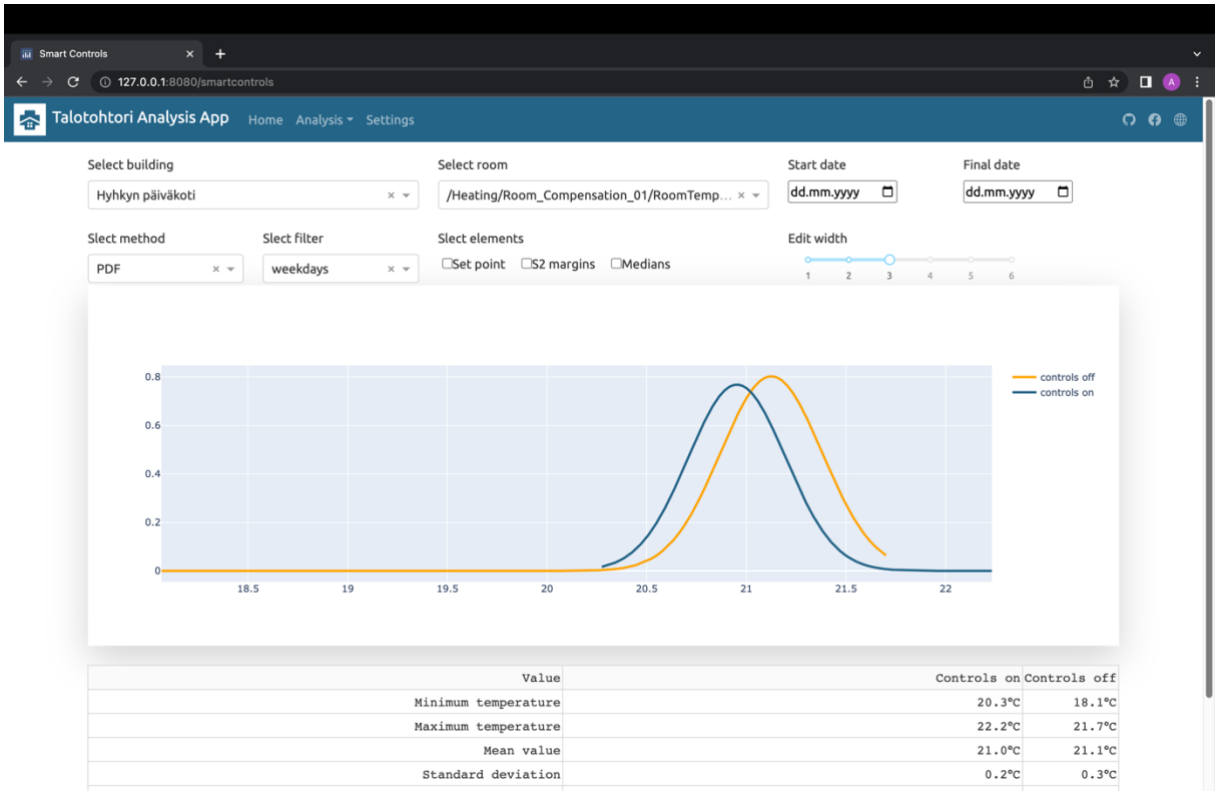
Figure 18. Plotly Dash home page



The project background is briefly explained on the home page. Bootstrap's page card components are used to organize the info on the page.

Eight widgets are on the smart controls page. Building, room in the building, plotting method, and time period are all selectable via four dropdown menus. To choose the start and end dates, there are two date pickers. Also, there is a checklist for adding and deleting extra chart components. Moreover, there is a slider to change the line's width. (See Figure 19. Plotly dash smart controls page)

Figure 19. Plotly dash smart controls page



Also, the project offers a Data Overview page where users can view a multi-select graph to examine the data and its quality. Certain room and area environmental conditions can be studied using the Rooms and Areas tab. Seeing the distribution and statistics of the environment condition variable is possible on the environment page. A link to the settings page, which was not completed, is used to view the error page that appears when a requested page cannot be found. Testing page was used to evaluate the appearance and functionality of specific components.

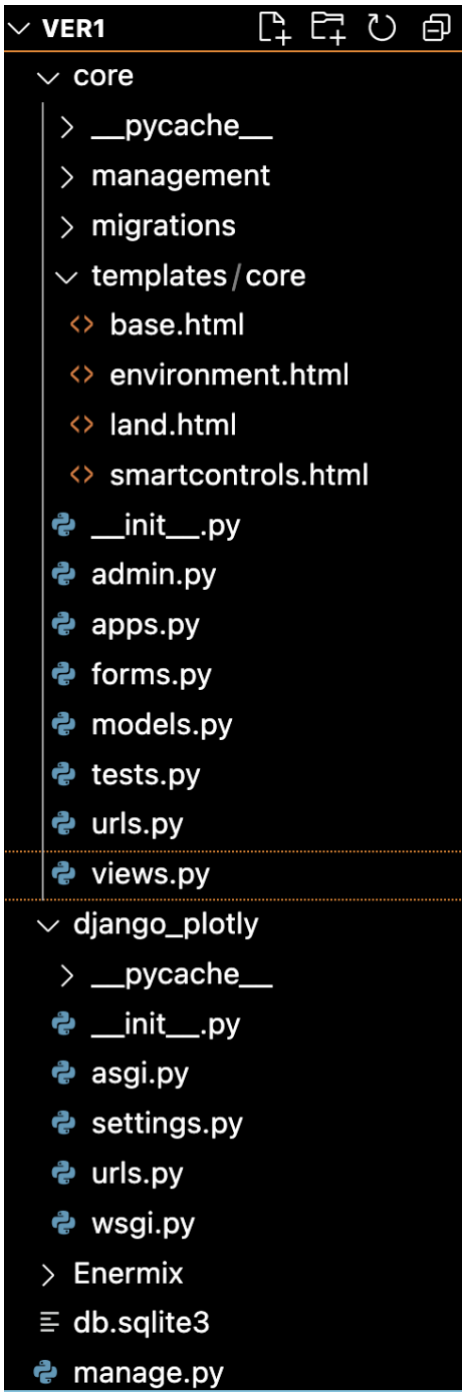
5.3 Django and Plotly

For this solution, separate virtual environment was created in which django project was launched and several libraries, including plotly and pandas, were installed.

5.3.1 Code structure

The project consists of a primary program file (‘manage.py’), a data folder (‘Enermix’), a django project folder (‘django_plotly’), and a django app folder (‘core’). (See Figure 20. Django and Plotly Project tree)

Figure 20. Django and Plotly Project tree



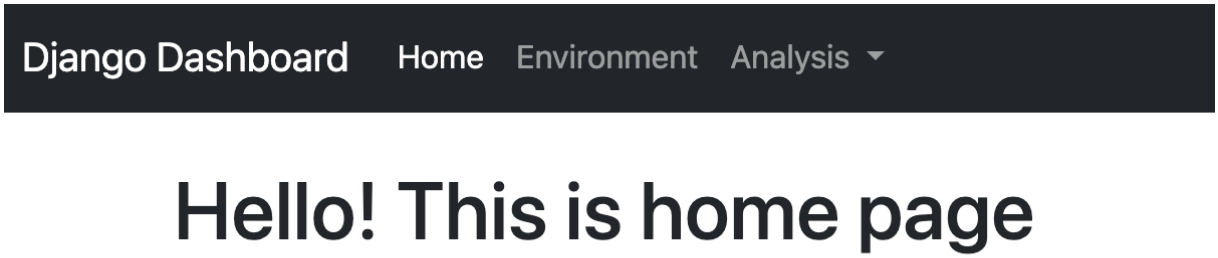
Commands are used to establish the complex Django project structure. Four scripts will be added to the directory after the project is launched. All project settings are contained in ‘settings.py’. Here applications are registered and the location of static files and database settings are set. ‘urls.py’ defines the relationships between url addresses and views. Although all of the url settings might be in this file, it is typically broken up into multiple sections. Communication between your Django application and the web server is established using the script ‘wsgi.py’. The ‘manage.py’ script is used to launch the debug server, operate with databases, and construct apps.

A second command is entered in the console to create an app in the project. This command will make a new folder and add files to it. The majority of files are named based on their purpose, e.g. controllers and views should go in views.py, models should go in models.py, tests should go in tests.py, admin settings should go in admin.py, and application registration should go in apps.py. And most of added files already have some template code for dealing with the aforementioned objects.

5.3.2 Visualization and functionality

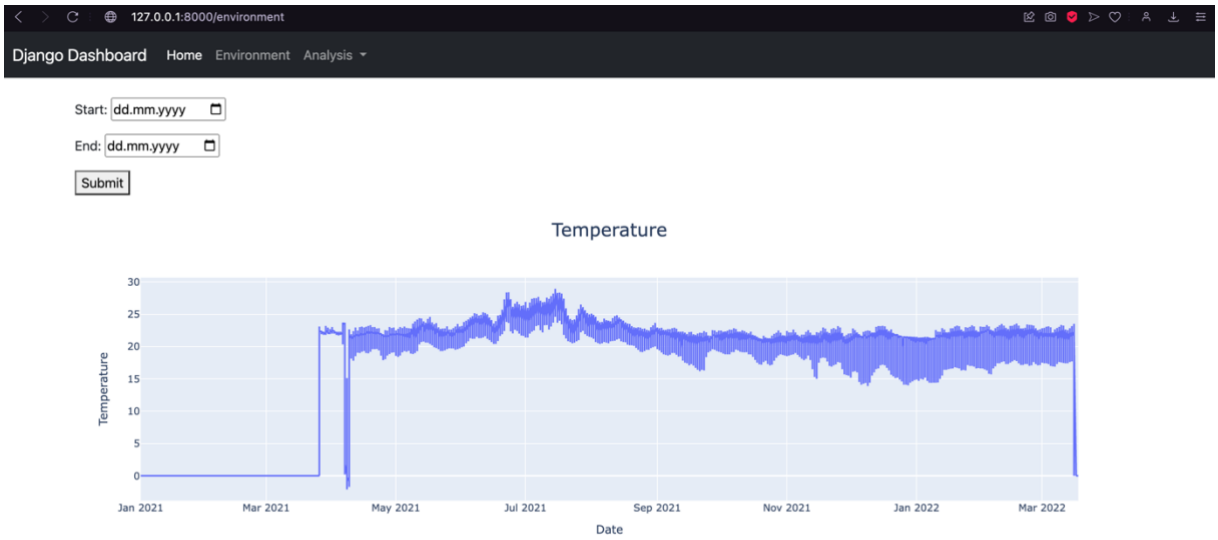
Only straightforward visualization was possible due to the intricate structure and data processing, as well as to a lack of knowledge and time. The app features two pages and a navigation bar on each of them. One title appears on the home page. (See Figure 21. Django and Plotly home page)

Figure 21. Django and Plotly home page



The Environment tab features a form with a submit button and two date selectors to choose the start and end dates of the data to be shown. A graph showing the temperature of a certain building can be found below the form.

Figure 22. Django and Plotly graph



6 Analysis

A crucial stage in ensuring the data is appropriately interpreted and used in the most meaningful way, is the result analysis. In the following section the results are broken down into smart controls performance and dashboards performance for analysis.

6.1 Smart Controls Performance

By analyzing the performance of the smart controls and the environment conditions in which they operate, we can better understand the effectiveness of the control system and determine where improvements can be made. This step is important, because it leads to better decisions by comprehending the data and how it can be used to guide decision-making.

Developed dashboards and visualizations provide various important insights. The distribution graph demonstrated how smart controls enhanced the temperature conditions in the buildings by bringing the temperature values closer to a specific set point value. This is depicted on the graph in Figure 19. Because of the narrower temperature distribution graph, the values are closer to the set point. That also implies that readings are within the permitted temperature range, neither above nor below.

The dependence of supply machine temperature on outdoor temperature was also demonstrated and represented by a box plot chart. Chart visualizes dependency in a clear form, which makes the data understandable and more human-readable.

Additionally, it was discovered that extremes in outdoor temperature lead to a broader range of internal temperatures, which might means that some sensors are located where doors and windows are regularly opened. But it also means that temperature in the buildings are less stable during peaks and dips of outdoor temperature.

It was discovered that the pressure difference and daytime are related. The pressure difference is substantially greater on weekdays and during office hours (from 8 am to 4 pm) than it is on weekends and outside of office hours (from 4 pm to 8 am).

6.2 Dashboards performance

A dashboard makes it simple to swiftly visualize data, empowering businesses to take wise decisions. But not every dashboard is made equal. To make sure dashboards are offering the maximum value, it is crucial to evaluate their effectiveness.

The PyWebIO dashboard is the solution that was most thoroughly implemented but also took up the most time. Although the visualization is straightforward and user-friendly, not much could be done to improve the dashboard's design and style. There were no navigational problems, but tab navigation is less practical than a navigation bar. This solution has a simple code structure, as well as a callback structure, which makes development fast and effective. A separate callback for each widget can, unfortunately, overwhelm the code structure. Development was made considerably simpler with the pandas package, which is used for data handling. Side platforms can host PyWebIO applications, PyWebIO does not offer its own server. It can be said that the PyWebIO library's full capability was utilized, and the analysis was clear and intelligible.

The Plotly Dash dashboard more closely resembles a website, as it features a proper home page with a simple-to-create design. It is easy to grasp and straightforward. The navigation bar, which also features link icons and a suitable logo, manages navigation. Although the code structure is not overly complex, it does necessitate familiarity with HTML and CSS. However, Bootsrap components enable complete design freedom. Data handling is broken down into three steps: reading data with pandas, storing data in a bowser session with plotly store, and accessing data with pandas. Plotly also provides the option to deploy your apps via their platform. This solution was not implemented to its full potential, but mostly because Plotly has much more potential than needed for this project.

Django framework combined with Plotly did not perform as well as other solutions due to its complicity and amount of skills required to take it to its full potential. With this framework, it is feasible to construct fantastic dashboards that are well-structured and understandable, but doing so will require extensive understanding of data handling, HTML forms, OOP, UX, Django functions, and Web development in general. The developed app manages navigation via a navigation bar and creates a simple graph. The structure is comprehensible and intuitive. This app has enormous potential to develop into one with fantastic design and extensive features.

7 Conclusion

After carefully analyzing developed solutions, it can be concluded that the most benefitial input was done by PyWebIO solution, which had more basic design and more complicated graphs. However, Plotly Dash solution has more potential for development and greater functionality. Django solution is more suitable and appropriate for development of web sites and other platforms, rather then dashboard creation, where modularity and code simplicity are demanded more.

The development of dashboards and analysis performed has been a successful endeavor that has allowed us to gain insight into the data and trends we were looking for. The dashboards have provided useful visualizations and the analysis has enabled us to gain a deeper understanding of the building data. We have been able to observe correlations and trends that were not previously visible, as well as gain an understanding of the data more quickly and efficiently than with manual analysis. The dashboards and analysis have been a beneficial to decision-making process and have provided valuable insights.

References

Few, S. (2022). *Stephen Few on Data Visualization: 8 Core Principles: Tableau Software*. Retrieved from Tableau Software Web site: https://www.tableau.com/blog/stephen-few-data-visualization

Hayes, G. (2019, August 25). *Which Programming Language Should Data Scientists Learn First?: Towards Data Science*. Retrieved from Towards Data Science: https://towardsdatascience.com/which-programming-language-should-data-scientists-learn-first-aac4d3fd3038

Calzon, B. (2021, November 3). *23 Dashboard Design Principles & Best Practices To Enhance Your Data Analysis: The datapine Blog*. Retrieved from The datapine Blog: https://www.datapine.com/blog/dashboard-design-principles-and-best-practices/

Hillier, W. (2022, December 7). *Blog: CareerFoundry*. Retrieved from CareerFoundry Web site: https://careerfoundry.com/en/blog/data-analytics/big-data-tools/

Simplilearn. (2023, February 24). *What is statistical analysis: Simplilearn*. Retrieved from Simplilearn: https://www.simplilearn.com/what-is-statistical-analysis-article

Hwang, J. (2020, July 28). *Plotly Dash vs Streamlit — Which is the best library for building data dashboard web apps?: Towards Data Science*. Retrieved from Towards Data Science: https://towardsdatascience.com/plotly-dash-vs-streamlit-which-is-the-best-library-for-building-data-dashboard-web-apps-97d7c98b938c

Khatri, V. S. (2023, April 5). *Flask vs Django: Which Python Web Framework to Use in 2023?: Blog: hackr.io*. Retrieved from hackr.io Web site: https://hackr.io/blog/flask-vs-django

Developer Survey: Stackoverflow. (2022, May). Retrieved from Stackoverflow: https://survey.stackoverflow.co/2022/#most-loved-dreaded-and-wanted-webframe-want

Ng, J. (2018, November 29). *Top 6 Tool Types For Data Analysis / Data Science - Save hours by using the right tool*. Retrieved from https://www.youtube.com/watch?v=23QtdnfhBRY&t=722s

Langmann, K. (2023). *The 5 Best Spreadsheets Software in 2023: Spreadsheeto ApS*. Retrieved from Spreadsheeto ApS: https://spreadsheeto.com/best-spreadsheet-software/

Best Self-Service BI Tools: SelectHub. (2023, April 13). Retrieved from SelectHub: https://www.selecthub.com/c/self-service-bi-tools/

Kumar, V. (2018, June 29). *7 Must-Know Programming Languages for Data Scientist & Data Analysts: Codemonkey*. Retrieved from Codemonkey Web site: https://www.codemonkey.com/blog/7-must-know-programming-languages-for-data-scientist-and-data-analysts/

Ryabtsev, A. (2023, April 10). *Web Frameworks: All You Should Know about It: Djangostars*. Retrieved from Djangostars Web site: https://djangostars.com/blog/what-is-a-web-framework/

Julia Micro-Benchmarks: JuliaLang.org. (2023). Retrieved from JuliaLang.org: https://julialang.org/benchmarks/

Appendix 1. Links to solutions

PyWebIO dashboard link:

https://github.com/hamk-uas/SmartCity-Dataviz-tool/tree/PywebIO

Plotly Dash dashboard link:

https://github.com/hamk-uas/SmartCity-Dataviz-tool/tree/main

Django dashboard link:

https://github.com/hamk-uas/SmartCity-Dataviz-tool/tree/django