Datenbanken und Web-Techniken / Databases and Web Techniques
Projektbeschreibung / Project Task
Sommer Semester 2023 / Summer Semester 2023

Introduction:

In today's world, good planning with digital tools is a great advantage, especially when it comes to private and business electricity planning. One approach is the photovoltaic system (PV/PS system). The problem is that it is often very difficult to calculate or specify in advance the usability and effectiveness of a photovoltaic system for a particular application in a particular place on the planet with a particular manufacturer. For this reason, it will be your task to create a basic possibility for better planning of photovoltaic systems based on digital tools.

Your task is to build a prototype of a web-based calculation system for a photovoltaic system with a specific manufacturer and product of your choice based on current weather data and local conditions.

Preliminary Remarks:

There is only one practical project task for all students in this semester. This task may be processed in groups of up to two participants, but it is not allowed to share solutions with other groups. The detection of significant solution parts shared between submissions of different groups is considered an attempt to defraud and marked accordingly for all involved students (independent of who shared solutions or who used them).

If there is more than one student in a group, the additional tasks mentioned at the end are obligatory (individuals may of course solve them, too). Also, the work should be divided evenly between both members and the indication of the authorship is required for all parts.

The subtasks may be solved by the usage of any database management system, any programming language, any frameworks, any libraries and any web service API techniques, i.e. there are no restrictions in choice.

Supported languages are German and English.

You will receive two marks for this subject, like listed in every study regulation. One for the project/presentation and another for the term paper!

Submisson:

The submission consists of ONE ZIP-archive with the name of the matriculation number(s):

1. a PDF-file of the term paper in paper format A4 and
2. a separate ZIP-archive (with a maximum file size of 10 MiB) that contains

2.1 the program sources (i.e. the source code and additional files like pictures or other resources that are required by the program),

2.2 a script to initialize the database (i.e. the submission of the database itself is not required) and

2.3 a small manual on how to use the sources to get a working web application (i.e. the submission of an executable program is not required).

Only one submission is required per group (but if you still insist on uploading the same project twice to OPAL, the latest submission will count for your group).

The submission has to be in full (including all parts) on time.

Please make sure, to upload your submission in time and do not exceed the file size limit! If you have problems with your file size limit, try to downsize your files, i.e. with reducers or delete unused code/node_modules/builds.

Task:

Term Paper:

A term paper is to be written, that satisfies the following conditions:

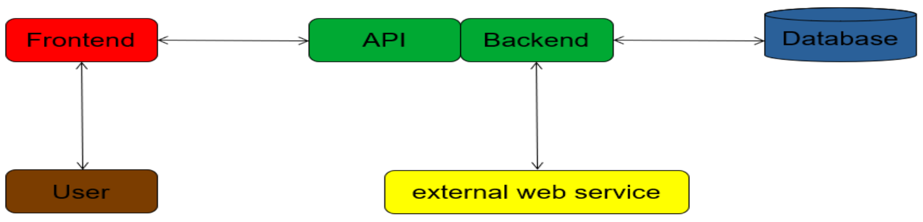
1. There is an amount of about 6 pages of content (i.e. excluding cover, index, lists, appendix, bibliography, API docs, etc.) by using default values for font size (12 px), line spacing and so on.
2. A good form and a balanced ratio of pictures and text is kept (i.e. just including dozens of screenshots without referring to them in describing text is bad form and may also rarely be counted as content).
3. On the cover page, there is recorded the name, study course and matriculation number of all participating students.
4. An overview of all utilized technologies (i.e. DBMS, programming and other languages, frameworks, web service API implementation, etc.) is given along with a short motivation, why they were chosen to solve a certain subtask. These technologies are also classified in the context of the lecture.
5. The project is presented in such a way that, after reading, you know all parts of the program and their functionalities without having explicitly executed them.
6. All used sources, libraries and technologies are referenced. There are no restrictions on reference or citation style (but styles should not be mixed). There is no need for real scientific writing (i.e. citing scientific papers), but it is still forbidden use content or quotes without giving references.
7. The appendix includes a complete web service API documentation (i.e. just listing examples is not sufficient) containing:
   a. a list of all endpoints and methods of the web service application programming interface,
   b. for every endpoint the list of the parameters and the return values each with type and meaning and the available status codes and
   c. for complex structures also the inner structure is to be documented respectively.
   d. The API documentation may be created using some tool, but it should fit with the other parts of the term paper in form and style (at least a bit, so there might be the need for reformatting things).

The term paper will be your first mark!

Programming:

The program consists of

1. a database for storing the data,
2. a backend for processing the data and interaction with the web service,
3. a web service application programming interface for providing the data and
4. a frontend for displaying the data and user interaction.



Database

1 All data is stored in the database, i.e. no additional storage (like files or web storage) is used, with the following exceptions:
1.1 Program configuration like database connection data or university user account data may be stored in configuration file(s).
1.2 Additional temporary storage is allowed within the workflow, but need to be cleaned up after user interaction is finished.
1.3 The frontend must never communicate directly with the database.

Disclaimer:

Storing binary data like random files in a database is considered bad practice. So please be aware that you should not do something like this in a real world scenario. This is just the usual academic task that lets you try something one time and meanwhile you hopefully learn, why this is a bad idea.

Backend

1 The backend processes the data.
2 It handles all communication with the database.
3 It provides the web service API to be accessed by the frontend.
4 It handles the communication with the external web services, for saving your own weather data:
4.1 The web service is used as intended.
4.2 You can save your weather data in your own format, i.e. map the external weather data schema towards your own or add/delete/modify attributes.
4.3 Write an cronjob, which is scraping the weather data every night (can be on development mode every time, when the server is starting).
5. The email process is handled here, also the sending process of the email.

Attention:

The external weather API's offer different schemas of weather, different authentication, different time interval, different system of UV/angle/clouds, accessibility in terms of calls per minute or day, please always follow their rules and restrictions!

Examples of weather APIs: Meteomatics (Meteoblue), Open Weather Map, Accu Weather, etc.

Web Service Application Programming Interface

1. The web service application programming interface provides communication between the frontend and the backend as well as the database.
2. It delivers the data from the database through the backend to the frontend.
3. It provides the user input from the frontend for processing in the backend.
4. API access failures are caught and responded with meaningful error messages (and possibly corresponding HTTP status codes).

Frontend UI

1. The frontend is a web application to be accessed by a web browser.
2. All common web browsers are supported, i.e. Mozilla Firefox, Apple Safari and Chromium derivates.
3. It is a responsive web application, i.e. different device and orientation types and mobile widths/heights are supported.
4. Usability aspects are taken into account, i.e. a nice user interface is created (UI), which leads to a good user experience (UX), so for example, if a user has to click somewhere, scroll right and down, and finally click another position to solve a task that could be achieved by just one click, then it could be considered bad usability.
5. The design is appealing.
6. Responsive web design is taken into place, mobile first is a nice to have.

Frontend

1. Users can create an account, update and delete an account
2. Users have a profile page.
3. Users can create, update and delete a project for their system.
4. Users can specify multiple photovoltaic system products within of a project.
5. User can choose between 3 different pre-defined photovoltaic system products. These predefined values should be realistic, look for the values of the companies, i.e. Hanwha Q-Cells, First Solar, Aiko, JinkoSolar, etc.
6. Users have an overview page over all their projects and the related photovoltaic system products.
7. Users can see a visual map (leaflet, openstreetmap, etc), where all the locations of their photovoltaic system products is located.
8. Users can generate a report for each photovoltaic system product, showing the electricity produced over the last 30 days is generated, based on 6 parameters.
9. The 6 parameters are: Power peak, orientation(N/E/S/W), inclination/tilt, area (m^2), longitude, latitude.
10. These 6 parameters must be pre-set each time a new project/product is created.
11. Be aware, that the peak power is not a constant variable, it changes every hour and every day! This is your part to face this challenge and give a good calculation process!
12. Users receive an email with the result of the report per photovoltaic system product
13. The weather data source for the report calculation is a live weather data.
14. Once the report is generated and the email is sent, the entire project is set to a read-only status.
15. Once the report is read-only, a user can view the results of the electricity output in an optical chart/diagram (data can be used from the email).
16. Users can filter for their active and old projects.
17. The report generation process is triggered either from a user or automatically after 30 days from the creation process of the photovoltaic system product.

18. Users can easily switch between their projects.
19. Users can use all CRUD operators on the visual map, that means, on a specific project where you have photovoltaic system products, you can view them, update and delete them and insert new products inside the visual map!

Additional Tasks for Groups

1. In the frontend there is an interaction for users to sync the weather data immediately, instead of every night with the cronjob, but the cronjob still exists.
2. Companies can create/update/delete their company accounts and can create/update/delete their associated photovoltaic system products. All photovoltaic system products from all companies are of course later available to all users later in the project/product creation process, that means the 3 different pre-defined photovoltaic system products are not given in the group task, the companies have to create them before.
3. There are 2 types of users, free users and unlimited users. The difference is, that free users can only create 1 project with 3 products/locations and unlimited users can create unlimited projects with unlimited products/locations.
4. An authentication layer is added towards the API communication, i.e.: JWT, Bearer, Oauth or API Key. Please mark this option in your API documentation!
5. The report date range option is a date picker, at least 1 day up to 1 year in the past
6. The default value for an report date range is still 30 days.
7. The user data and explicitly the user password is encrypted.
8. The process of deleting a user account is a soft delete process. In addition, an API endpoint is available to show all soft deleted users.
9. The process of calculating for the rate of PV/Watts per day from the photovoltaic system includes more parameters, instead of 6 parameters you have 8. Additional parameters are clouds and system loss (default 14%).
10. The user should be able to enter any location on the planet and your application will convert the location (i.e. Chemnitz) to a longitude and latitude value for a new product at that location!

Examination

The examination consists of a 15-minute presentation, which should meet the same criteria as the content of the term paper, but focus is on the live demonstration. The API documentation is not presented here.

The project and the used technologies and an introduction into the topic should be presented. The main part of the presentation is to present your features and funcionalities, so include a live demonstration of the project or a demonstration video demonstrating all parts of the practical task. For group work, the presentation time should be divided equally between both students.

Afterwards, some questions are given, which primary focus on the project and the term paper, but may also cover the lecture and exercise.

Finally, there will be a short consultation and you will be informed about your second mark for the project and your presentation.

Dates

Handout of task description:

- starting 2023-05-25 16:00 CEST (UTC+2) (25.05.2023 – 16:00)

Submission of project:

- until Saturday night: 2023-07-01 11:59 PM CEST (UTC+2) (01.07.2023 – 23:59)
- via OPAL:
- https://bildungsportal.sachsen.de/opal/auth/RepositoryEntry/297435137/CourseNode/1654050630382772004

Oral exam and presentation:

- between 2023-07-04 (04.07.2023) and 2023-07-14 (14.07.2023)
- appointment allocation will be done via OPAL
- examination will take place in person, the room will be announced in OPAL