Project Databanken en Webtechnologie

Tom Godden - tgodden@vub.be

2023-2024

Read this document at least twice so that you know what you have to do! For this project, you will design a website for selling event tickets. Your deliverables include the database, the RESTful API application and the site's frontend application. In addition, also a report and a presentation is required.

1 Overview

Your site needs to have at least the following basic functionalities:

1.1 Ticket Sales

- The site needs to have a functional interface and use your RESTful API to communicate with your database.
- The site needs to display upcoming events, including the artists or bands, date, time and the location of the event, a poster image, the price of the tickets, the amount of tickets remaining and the event host.
- Events should be able to have multiple tiers of tickets, e.g. front row tickets, seating tickets, ...
- The events should be somehow categorised (e.g. pop, metal, comedy). Users should be able to filter the events based on the category; the filtering should happen at the server's side.
- You should be able to hold sales for events. The site should display when an event is on sale.

1.2 Users

Users need to be able to create an account and log in to the website. Use cookies to store the user's session.

There are two kinds of users: **customers** and **event hosts**.

1.2.1 Customers

- A customer has a shopping cart they can add tickets to.
- A customer can "purchase" the items in their shopping cart. You don't need to use an actual payment system, it's enough for a user to see the total price and be able to press "confirm".
- Customers should have a shopping history linked to their account, displaying the tickets they purchased.

1.2.2 Event Hosts

- Event hosts have one or multiple locations linked to their account.
- Event hosts can add an event to the site by entering the necessary information.
- Event hosts can see all events they are hosting or have hosted, with an overview of the total number of tickets sold for these events.

2 Optional Extra Requirements

You are encouraged to extend your application with extra requirements. Here is a non-exhaustive list of extensions to give you some inspiration:

- **Calendar Export**: Your site can export all events to a calendar (e.g. .ics) file format.
- **Administrator Account**: Add an administrator account that can remove other customer and event host accounts, but not other administrator accounts.
- **GDPR:** Add two calls to the API. The first API call returns all data that is linked to the user executing the call. The second API call deletes all this data. Note that bookkeeping of shows should still be correct!

3 Project Details

The project is divided into four parts. Each part is needed for the next one, so make sure that you start on time. You *need* to hand in each part to be graded for the project.

Below are the deadline dates for each step. **Each step has to be handed in before this date, so if the deadline is October 25th, you can hand in the project up until October 24th 23:59!**

Subject	Deadline
ER Diagram	In Class
Database	25 October
RESTful API	8 November
Final Product	22 November

You are allowed (and encouraged) to make improvements to previous steps, but you need to clearly document these changes. If you do make changes, include the changed files in your next deadline and explain what changed!

3.1 2.1 Reports

Important: for each step, you will also have to hand in a report as documentation and to describe your reasoning and development process. Your report gives an overview of how your code works and how your database is structured. This does not mean that you have to literally copy your code to the report; instead you give a high-level overview of the design of your project and any design decisions you made. A design decision could be why you decide on a certain database design, why you use or don't use specific parameters for your API calls, etc. We want to see you think about this project and understand your mistakes, not simply type in code until it works!

3.2 ER Diagram

Deliverables: ER Diagram and documentation

In class, you will learn how to design a database by using ER diagrams. Create an ER diagram to get an overview of what your database should look like. Think about the requirements of the applications and how you will implement these in the database.

Alongside the ER diagram, you have to hand in a small report that describes the ER diagram and explains your reasoning for its design.

3.3 Database

Deliverables: Database, report and example code
Create your database, based on the ER Diagram, and add some data to it. In addition, design the queries your application will use, and demonstrate them in a well-documented program. The documentation can be in the code itself, or can be in a separate file like a Jupyter NoteBook.

3.4 RESTful API

Deliverables: Server code, API documentation, report, example client code
In this step, you will design the RESTful API for your program and demonstrate the communication between the front-end and back-end. Once again, keep the requirements of the application in mind when designing your API.

- You should create a stand-alone server application that hosts the RESTful API, and create a documentation for this API.
- Demonstrate its workings with some well-documented example client code. This code should call the API using HTTP.
- Create documentation for this API aimed at potential developers using your API.
- In your report, describe and motivate the structure of your system, and explain the test cases.

3.5 Final Product

Deliverables: Database, API server code, Web server code, website documentation, API documentation, report
For the final step, you need to design the website and have it communicate with the previously designed API. Your site should use HTTP calls to get the data from the API, from which it will generate HTML (and CSS) code. *This means that the API and the Website are two distinct stand-alone server applications.* Write a document that demonstrates the use cases of the site and include pictures to demonstrate this. This document can be a

slideshow, but make sure that you hand it in as PDF. In your report, describe the structure of the final product (use diagrams where possible) and what functionalities your application has. Describe what design choices you have made to get to this final product. Give a brief overview of what parts of the design process were difficult and how you overcame these difficulties. Finally, provide a conclusion on the feedback of this project/course.

4 Hints and tips

- Use unit tests for easier debugging. Unit tests are also very useful to make clear examples! More information on unit testing in Python can be found at `https://docs.python.org/3/library/unittest.html`
- If you are stuck or have questions, ask your favourite search engine. Knowing how to find answers on the internet is a big part of programming. If you're still stuck afterwards, ask the teaching assistent.
- Although a beautiful site is nice, it is not a requirement for this project, so do not waste too much time on creating award-winning interfaces. A clear interface is more important than a beautiful interface.