BUK Reservation System

Artem Kuznetsov
ČVUT–FIT
kuznear1@fit.cvut.cz

May 26, 2024

My assignment was to make a reservation system that creates events in Google Calendar for the Buben student club to work with the club's information system (IS).

1 Introduction

Many student union clubs have rooms and services that can be reserved and booked by their members. Unfortunately, because bookings are not automated and mostly done manually through e-mailing, club rooms and services are not as popular as they could be. I am eager to automate this process so that members would no longer have to spend days emailing managers, and the only confirmation that might be required is permission from the dorm manager for events with large numbers of people. This will also simplify the work and responsibilities of those in charge of club facilities, due to high compartmentalisation of existing reservation methods and a constant need of tracking each. I am aiming to consolidate all of this into one web application that would be easy to use for both regular users and managers. This is a big project, which will require a lot of time to develop and design properly, so far I have managed to implement the basic aspects of the web application.

2 Reservation System

The original idea was somewhat different. In the new Bubeneč dormitory, almost every door has an RFID card reader. As one of the club chairmen and head of the section dealing with club rooms, I would like to provide access to the booked rooms directly onto ISIC cards registered within the dormitory and Club's information system (IS) in the future. Fortunately, this will be possible to implement, but at the moment the solution requires RFID reader system provided by ČVUT to be connected with IS and is not yet ready. Therefore, this idea has been replaced by a more realistic goal: to automate the booking process and create a foundation for implementing the original concept.

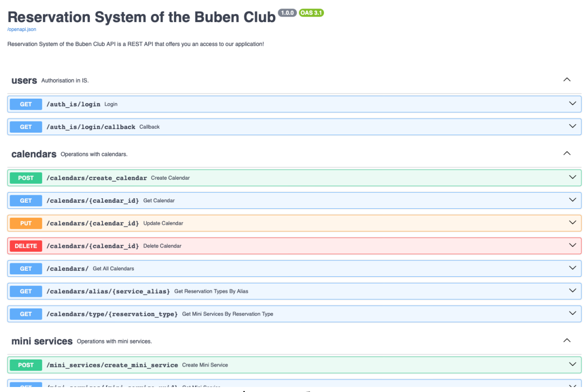
Figure 1: RFID reader

Moving on to the application itself, the main task I see is: an application that allows the user to book a room or service themselves. A common problem is that people forget, do not understand or ignore the terms and conditions of how and when you can book, so the app would automatically restrict users according to the rules, for example – you can not book at night or need to book a day in advance. The booking should not be extensive, so I came up with the idea of linking our booking system to the club's information system called IS, which is where all the user information will be taken from. The Club's information system (IS) contains details about club members, streamlining the reservation process by allowing the app to access their data during the booking form completion. This enables the app to automatically check members' roles, available services, and status, eliminating the need for users to repeatedly enter their information. There are three types of member roles — regular, active, and manager — each with distinct capabilities and limitations.

Here is the basic idea behind the app and its features: Users can select a specific service, choose the type of booking they want, and fill out a form to create an event in Google Calendar. If there are any restrictions, the app will display a warning indicating the issue. If everything is in order, the booking will be confirmed, ensuring that the room at the club will be available for the user at the designated time.

3 Used technologies

As I mentioned before, I use the API of IS to get information about users, their roles, and so on and Google Calendar to create events in the calendar, at the moment it is not used for anything else. The basis for my application is the FastAPI framework for building APIs with Python based on standard Python type hints.



Reservation System of the Buben Club 1.0.0 OAS 3.1

/openapi.json

Reservation System of the Buben Club API is a REST API that offers you an access to our application!

users Authorisation in IS.

```
GET     /auth_is/login  Login
GET     /auth_is/login/callback  Callback
```

calendars Operations with calendars.

```
POST    /calendars/create_calendar  Create Calendar
GET     /calendars/{calendar_id}  Get Calendar
PUT     /calendars/{calendar_id}  Update Calendar
DELETE  /calendars/{calendar_id}  Delete Calendar
GET     /calendars/  Get All Calendars
GET     /calendars/alias/{service_alias}  Get Reservation Types By Alias
GET     /calendars/type/{reservation_type}  Get Mini Services By Reservation Type
```

mini services Operations with mini services.

```
POST    /mini_services/create_mini_service  Create Mini Service
```

Figure 2: App documentation

4 Models

I initially planned for the reservation calendars to be static, without allowing new additions. However, I realized the program needed to be more adaptive, leading me to rewrite a significant portion of the code. Now, the program can handle new services or club rooms and changes in reservation rules without requiring further rewrites. Managers can create and delete calendars within the app and set rules for them, all linked to existing Google Calendar IDs. This ensures flexibility and ease of management as the club evolves.

A similar idea is behind the mini services model. Currently, mini services exist only as additional services to provide the user with a choice when reserving the main premises, access to which will be given by the manager.

The user model primarily includes details such as the username, token, active membership status, and the rooms they manage. It serves as a communication tool with the IS and for verifying permissions for various actions within the application.

5 Algorithms

Regarding the algorithms in my program, I did not use any complex ones. The most challenging part was implementing the checks required to create events based on the user's role in the club. Additionally, ensuring correct creation, updating, and deletion of database objects was crucial so that all related objects are updated appropriately. I utilized basic CRUD methods and supplemented them with specific methods to retrieve objects by certain parameters.

6 Encountered problems

While writing the program, I encountered several problems that I would like to describe in this block.

Starting with the objects I have in my program, specifically calendars and mini services. Right now they are linked together in a odd way, namely by a list of strings of mini-service rows in the calendars and a common service. In the future a list of objects in each of the models will be created, but there was a problem with the implementation of the many-to-many relationship and subsequent testing, so I decided to leave it as it is, with the intention of finding a better way to link them to each other. Now I can think about creating a service model, which can allow many calendars and mini-services, but even here there are pitfalls that does not allow to call it the best solution at the moment. Similar situation with collision with calendars in a calendar, it would make a loop if it was a list of objects and the testing process would be very complicated, so now a list of row id of calendars collisions is stored.

Another challenge was implementing user authorization, as I had no prior experience in this area. I was unfamiliar with methods to maintain and properly update the token. Currently, to check user permissions, the username is sent in the header to retrieve the user object within the app. However, this approach is neither efficient nor secure and will be improved in the future.

I encountered the problem of identifying a user's role within the club. Initially, I intended to use tags, but IS do not have LDAP configured to retrieve and utilize these tags. Fortunately, I found a solution for room managers, as they are designated as administrators for the service (room) they oversee in the IS. However, the issue of distinguishing between active and inactive members remains. Currently, this is managed with a note in the IS labeled "active," which only a few users can change. In the future, this will be replaced by tags once the IS is properly configured.

And the last problem is the impossibility to make events in Google Calendar with the status tentative, - an event which must be confirmed by the manager, for example, after receiving confirmation for a large amount of people event from the head of the dormitory. Google no longer supports this feature, so my solution was that to create the event in a private calendar, where after the confirmation it will be manually assigned to the correct calendar and this method is certainly less elegant.

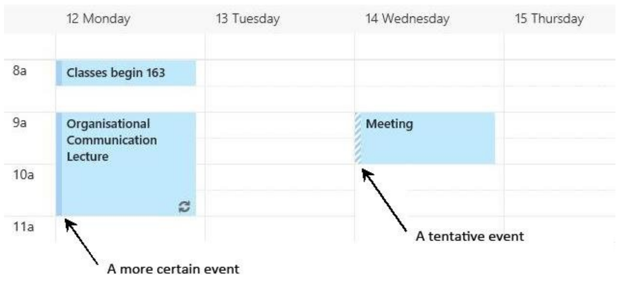
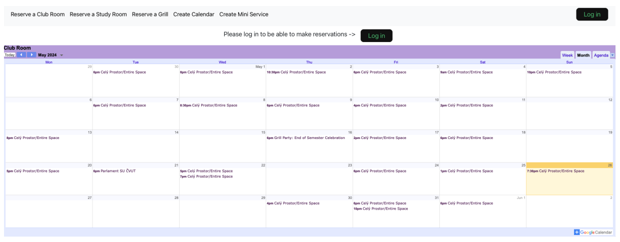


Figure 3: Tentative event

7 Results

As a result, I've developed an app allowing members to autonomously book club rooms, with manager approval required only in exceptional circumstances. Booking conditions are tailored to each user's role. This represents the foundational level of functionality achievable with current resources and capabilities. Looking ahead, addressing the aforementioned challenges and devising workarounds for the absent Google Calendar features will be key areas for improvement.

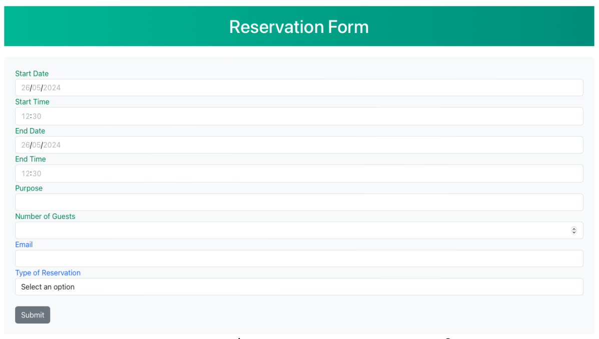


Please log in to be able to make reservations ->

Club Room

Mon	Tue	Wed	Thu	Fri	Sat	Sun
29	30 6pm Celý Prostor/Entire Space	May 1 8pm Celý Prostor/Entire Space	2 10:30pm Celý Prostor/Entire Space	3 6pm Celý Prostor/Entire Space	4 9am Celý Prostor/Entire Space	5 10pm Celý Prostor/Entire Space
6	7 6pm Celý Prostor/Entire Space	8 8:30pm Celý Prostor/Entire Space	9 6pm Celý Prostor/Entire Space	10 4pm Celý Prostor/Entire Space	11 2pm Celý Prostor/Entire Space	12
13 8pm Celý Prostor/Entire Space	14	15	16 6pm Grill Party: End of Semester Celebration	17 2pm Celý Prostor/Entire Space	18 6pm Celý Prostor/Entire Space	19
20 5pm Celý Prostor/Entire Space	21 6pm Parlament SU ČVUT	22 5pm Celý Prostor/Entire Space 7pm Celý Prostor/Entire Space	23	24 6pm Celý Prostor/Entire Space	25 1pm Celý Prostor/Entire Space	26 7:30pm Celý Prostor/Entire Space
27	28	29	30 4pm Celý Prostor/Entire Space	31 6pm Celý Prostor/Entire Space 10pm Celý Prostor/Entire Space	Jun 1 6pm Celý Prostor/Entire Space	2

Figure 4: App main page



Reservation Form

Start Date	26/05/2024
Start Time	12:30
End Date	26/05/2024
End Time	12:30
Purpose	
Number of Guests	
Email	
Type of Reservation	Select an option

Figure 5: App reservation form

8 Conclusion

In conclusion, there is still a lot to be implemented, but the solution we have now can already be used in test form. After that, only the development of the application itself remains, which I would like to touch on in more detail.

The most important thing I am aiming for is to create a fully automated solution where people can book club rooms and get an access to them on their ISIC cards.

I am planning for my app has to be highly adaptable, so in the future it could be augmented with all sorts of ideas to implement reservations of anything within the club, like renting board games for example. Best of all, it will all be in one place.

The last thing is to further integrate my app with Google Calendar so that you can create, modify, delete events and calendars from Google Calendar right in the app without having to additionally go into Google.

Buben

Figure 6: Buben club logo

References

[1] Calendar API documentation. online. [cit. 2024–05–25] https://developers.google.com/calendar/api/guides/overview.

[2] Conda documentation. online. [cit. 2024–05–25] https://docs.conda.io/projects/conda/en/latest/user-guide/install/index.html.

[3] FastAPI documentation. online. [cit. 2024–05–25] https://fastapi.tiangolo.com.

[4] IS API documentation. online. [cit. 2024–05–25] https://is.buk.cvut.cz/oauth_api.

[5] Pytest documentation. online. [cit. 2024–05–25] https://docs.pytest.org/en/latest/contents.html.

[6] NeuralNine. Google Calendar Automation in Python. online, 2023. [cit. 2024–05–25] https://www.youtube.com/watch?v=B2E82UPUnOY&t=456s.

[7] Noah Tarr. Size optimized conda environements. online, 2022. [cit. 2024–05–25] https://gist.github.com/NoahTarr/cdc0af59ebc84fc9d936eece35ebfaf7.