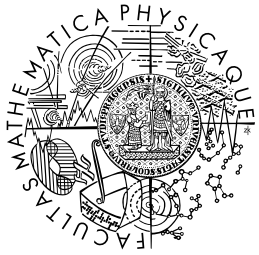

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Bc. Jakub Sýkora

Thesis title

Name of the department

Supervisor of the master thesis: Supervisor's Name
Study programme: study programme
Study branch: study branch

Prague 2023

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

Dedication.

Title: Thesis title

Author: Bc. Jakub Sýkora

Department: Name of the department

Supervisor: Supervisor's Name, department

Abstract: Abstract.

Keywords: key words

Contents

Introduction

1. Introduction

An example citation required for successful build: Example [20222]

1.1 Linked Data and Decentralized Data Stores

1.2 SOLID Specification

SOLID Servers and Pods

Authentication and Authorization

SOLID Applications

1.2.1 Present-day Calendar Applications

Google Calendar

Apple iCloud Calendar

Calendar Application Using SOLID Specification

1.3 Goals

G1 Develop calendar application using SOLID protocols

- **a)** Compare and utilize existing SOLID libraries
- **b)** Replicate and support behaviour of present-day calendar applications
- **c)** Support export and import of calendar data in iCalendar format
- **d)** Stored data will adhere to linked data principles

G2 To demonstrate the ability of the calendar application to operate in an ecosystem of interoperable SOLID applications, develop a sample application accessing the data of the calendar application.

2. Analysis

2.1 Calendar Features

Calendar Events

The essential part of the calendar is its events. Therefore, the user needs to be able to add, remove and modify calendar events. Furthermore, the application needs to display event data in a structured way to emphasize important information and hide the details. As we are creating a calendar application based on SOLID specification with an option to store and query any information as linked data, such structured information is describable by linked data vocabularies, making them available for further processing by other applications. For example, another SOLID application, such as a task manager, can display calendar information in a list of tasks. Therefore, we define the following requirement:

R1 Calendar events should contain structured data that can be consumed and queried by other applications.

The following sections describe event form with fields the application must support and introduce event features of present-day applications from section 1.2.1, which our application will re-implement.

Calendar Event Form

Each event requires a start date, event duration, and short text used as a title to display an event in a calendar visualization. In addition, many events occur at some physical or virtual location. Therefore, the application should provide fields for description of location information. The transformation of user inputs to linked data is relatively simple for data types such as time or title text. However, in the case of more structured data, such as an address with an arbitrary list of address parts, user experience can't be degraded by only providing a fixed number of fields. Therefore, the application will use an interactive form that would enable users to define information, such as address parts, in a user-friendly way.

R2 In calendar event form, users can define the following data fulfilling requirement **R1**

- Start time and end time of events with a date picker form, using current date in locale time zone by default, with option to explicitly set specific time zone
- Text field used as text title in calendar
- An optional address information of physical location, using an interactive form to autocomplete address parts
- An optional URL link to specify location of virtual meeting

To provide users with such an interactive address form, we would like to re-implement address autocomplete features offered by Google Calendar, described in section 1.2.1. However, to still support requirement **R1**, we will need to find an autocomplete service offering results to be further separated into specific address fragments. Figure 2.1 shows a mock of the address form we would like to use in our application. In addition to an autocomplete feature, a user could select specific location descriptors such as room and floor numbers. Thus we can use each address fragment from auto-complete functionality or currently specified address descriptors to describe address information as linked data.

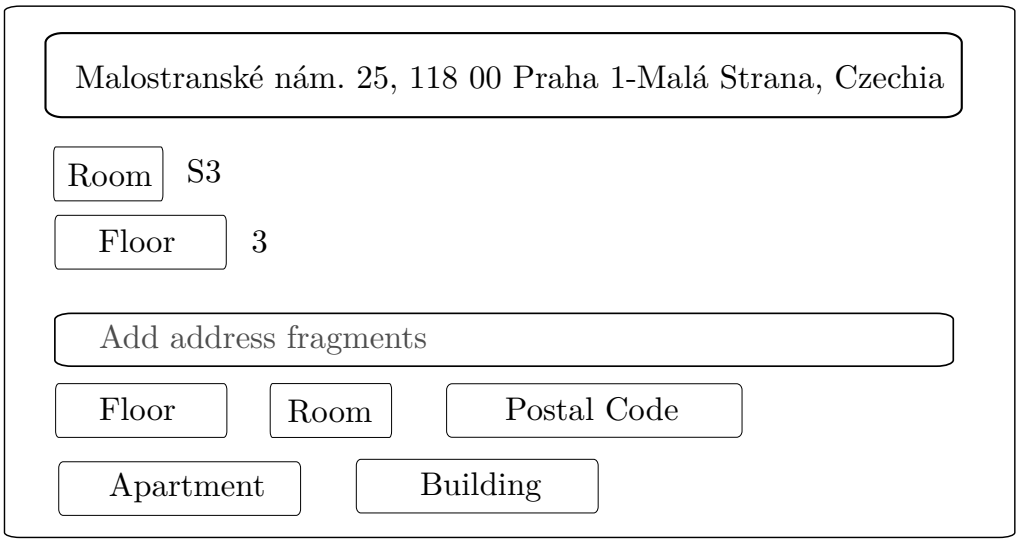

Figure 2.1: Mock of interactive address form

The address form needs to cover many specific scenarios. For example, users should also be able to specify addresses that can't be found by auto-complete functionality, even if such addresses can't be further separated into smaller fragments and queried by other applications. We will describe such specific requirements by the following list of use cases.

U1 Specifying address with no auto-complete result

Next to the data that can be structured, a user should be able to write any other information in a block of unstructured text. File attachments are an additional feature that would increase calendar usability. The calendar application should allow storing arbitrary attachments such as pictures, documents, or recordings.

R2 In calendar events, users can define the following unstructured data

- Optional description text
- Optional file attachments

Event Groups

As calendar users utilize a calendar for many activities such as work meetings, university lectures, or their free time events, users should be able to differentiate between each specific type of activity visually. The present-day calendar applications introduced in section 1.2.1 also support such event grouping. Therefore, users should be able to categorize each event into a collection we call an *event group*. Users can create an arbitrary number of event groups, each differentiable by group name and color. For example, figure 2.2 shows a calendar with two event groups for work and school events.

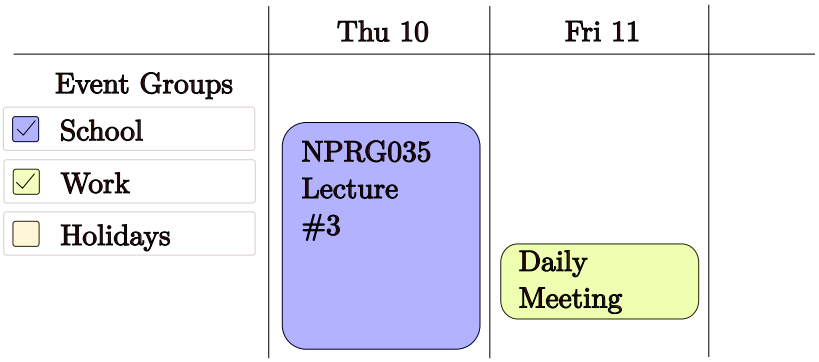

Figure 2.2: Example of two *event groups* displayed in calendar

To make application operations and user experience across the application more consistent, we define a rule that each event must belong to some group. If a user does not assign an event to some specific group, the application assigns the event to the default group. In addition, users can move each event group from one event group to another. In event group deletion, the calendar also removes all defined in the group.

R3 Users can assign events into event groups.

- Users can create and remove own event groups.
- Application assigns a new calendar event to the *default* event group if no user defined group is selected.
- Calendar events can be assigned from one calendar group to another.

Reccuring Events

R4 User can define *reccuring events*.

- Multiple future occurences of single reccuring event can be modified in single event update.

All-Day Events

Calendar Views

The application should support displaying calendar events in different calendar views, such as a view of calendar events in a single day, week or month. Views in

different timespans increase the application's usability by hiding non-important events or providing a valuable overview of future events. In the following subsection, we will introduce functional and non-functional requirements that each calendar view should support.

R5 Users should be able to display calendar events in a view displaying events of single day, week or month.

Shared Functionality of Calendar Views

In this subsection, we define a set of features that each calendar view should offer in the application. Defining such features increases the application's usability by defining expectable behavior shared in each calendar view. As we want to make the calendar application interactive, users should be able to modify calendar events displayed in a calendar view by simply moving the visualization of the event from one timeslot to another. By modifying an event in a calendar view, users can set a different start time for an event or change the duration of an event, as shown in figure [X], where the user *drag and drops* an event to a different location, changing start of the event.

R6 Users should be able to interactively modify start and duration of events displayed in calendar views.

R7 Users can create multiple instances of calendars view to display different calendar groups.

Sharing

One of the more advanced features of our application would be sharing calendar events with other application users. Users should be able to specify who can see and modify their events. The application should store user data only in SOLID Pod and follow other SOLID principles defined in the goal []. Therefore, Web Access Control, introduced in section [], will be used to implement calendar sharing features.

R8 User can create publicly shared calendar group that is readonly

R9 User can share calendar group only to group of users

- Calendar group is privately shared and readonly
- Calendar group is privately shared and modifyable by each user with access

R10 Users can create copy of other calendar groups that are shared with them

R11 Owner of calendar event can add participants to event

- Participants with access to the event can modify participation status

3. Implementation

3.1 Implementation Analysis

3.1.1 Linked Data and User Input

R1 can be implemented by using Google's Maps JavaScript API. Using Google's structured information about a place selected by autocomplete.

Conclusion

Bibliography

Example. Example. *Example*, 20222.

List of Figures

List of Tables

List of Abbreviations

A. Attachments

A.1 First Attachment