Local Greetings Documentation

(Bazon Bogdan and Hasan Mehedi)

Problem Description

Developing a Web application as a Social network for the city of Iasi, Listing public sport fields and allowing gatherings among people. Depending on the event, the joining criteria may be First Come First Serve or based on users with at least P points. A list of meetups will be generated to users as an RSS feed that will be eventually sent as an e-mail.

Allowed services: OpenStreetMap API.

Who are the actors?

- Guest User: Is only allowed to browse sport fields and events.
- Logged User: Can Create new events
- Administrator: Can manage created users, events. Can export, import data.
- Map Service: Provides the map of Iasi with locations of public sport fields

Use Case Scenarios

Guest Interactions:

1) User browses available sport fields
    - In home screen, the user clicks on see available sport fields
    - On this page, the user can see details about available fields
    - ALTERNATIVELY: The user can specify a string or sport field type and will display only the fields conforming with the search result
    - EXCEPTION: User inserts an invalid String or field type -> A message will be displayed accordingly
2) User browses available events
    - In home screen, the user clicks, browse events
    - On this page the user can see details of available events
    - ALTERNATIVELY: The user can specify a string or event type and returns the events conforming with the search result
    - ALTERNATIVELY: The user can see events by Public, Joined Or Created categories
    - EXCEPTION: The user selects Joined or Created tabs -> The user is sent to the login page
    - EXCEPTION: User inserts an invalid String or event type -> A message will be displayed accordingly
    - EXCEPTION: User tries to join an event -> Redirects to the login page
    - EXCEPTION: User tries to create a new event -> Redirects to login page
3) User logs in
    - In home screen, user selects the login button

- Here is presented a form consisting of email and password
- User fills the form
- User is sent back as a registered user.
- ALTERNATIVELY: The user can choose to register a new account
- EXCEPTION: User puts invalid characters -> A message will be displayed and asked to try again
- EXCEPTION: User puts invalid email -> A message will be displayed accordingly
- EXCEPTION: User puts the correct email but not the correct password -> A message will be displayed accordingly
- EXCEPTION: User logs in with administrator account -> Will be redirected to the admin DASHBOARD

4) User registers to the platform
   - In home page, user selects the create new account button
   - Here is presented with a form that asks for username, email and password.
   - User fills with credential
   - The user is redirected to the login page and user logs in with the newly added credentials
   - The user is redirected to the home pages
   - ALTERNATIVELY: The user can choose to log in instead.
   - EXCEPTION: User puts an email already registered -> A message will be displayed accordingly
   - EXCEPTION: Invalid character or missing form entries -> A message will be displayed accordingly
   - EXCEPTION: Password and Confirm Password do not match -> A message will be displayed accordingly

Logged in User Interaction:

1) User browses available events

- Same interaction as the Guest user case.
- ALTERNATIVELY: The user can join Events
- ALTERNATIVELY: THe user can check Joined and Created events
- ALTERNATIVE: The user can join or leave event
- EXCEPTION: User created event -> Will not be displayed on public or joined event tab, and will automatically join. Also the user has the option to edit events on this tab.
- EXCEPTION: The number of participants reached: An rss feed is generated consisting on event detail and a list of participants

2) User creates an event
   - On Event page, the user selects the create event option
   - Here is presented a form and a map pointing to available sport fields
   - User fills the form
   - The user is redirected to the event page with newly created event
   - EXCEPTION: Invalid credential -> A message will be displayed accordingly

3) User updates his profile
   - On home screen, the user selects the profile option
   - On this page, the user has option to go to the events page, update profile or See a list of previously participated events
   - User selects edit profile
   - Is presented With a form consisting of Username, email and password (The password is optional).
   - User fills the prompt
   - Is redirected to the user profile page
   - EXCEPTION: Same from Register page -> A message will be displayed

- EXCEPTION: No password is given -> The updating procedure continues as normal

4) User updates an event
    - On events page, the user selects the created events tab
    - On events created by the user, has the option to edit the event
    - User selects the edit event
    - Here the user is given the same form as in Creating the event form and a delete button to delete the event
    - User fills the form
    - User returns to the event page with the modified event

Administrator Interaction:

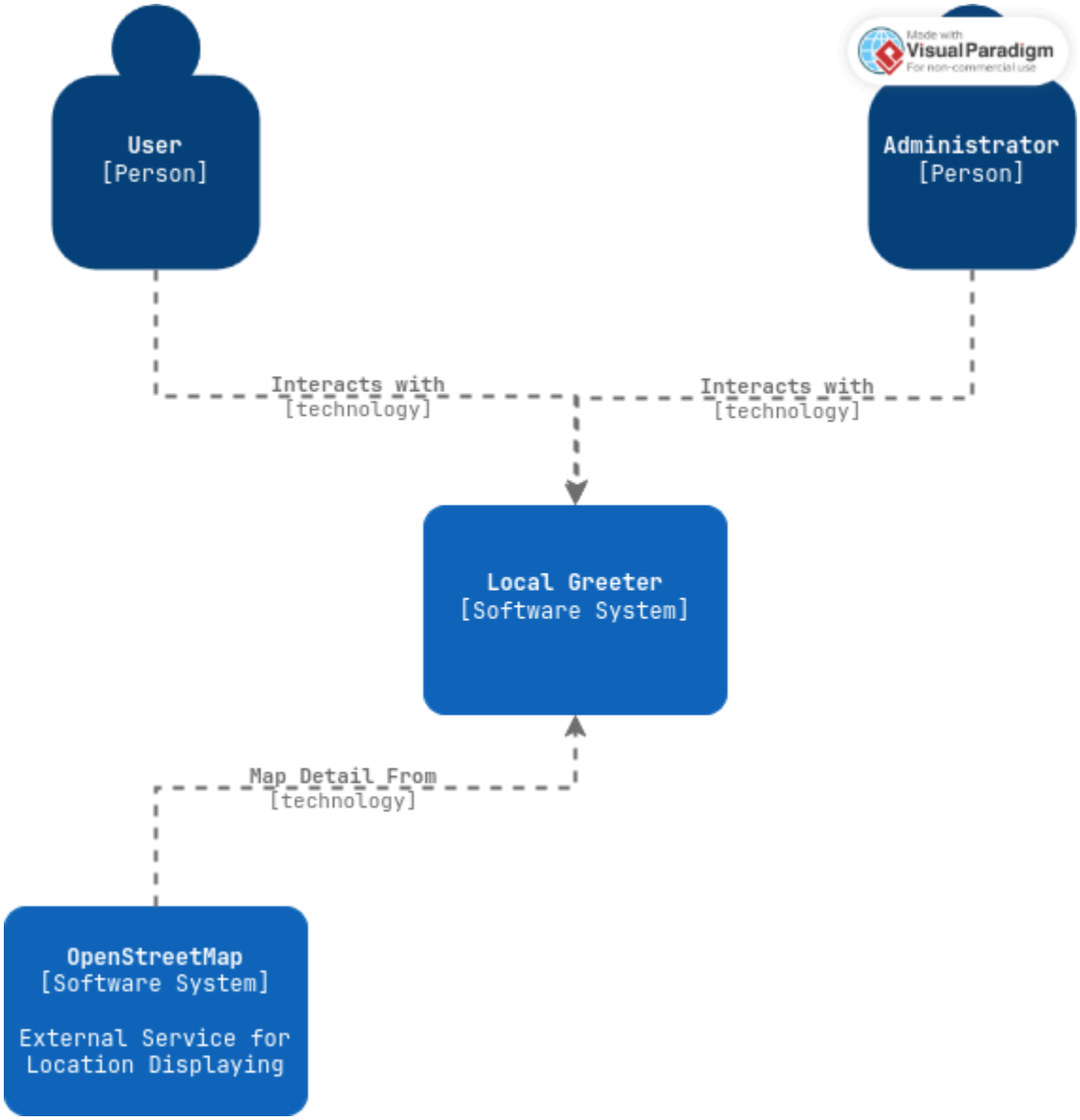
1) Admin deletes an user
    - On admin DASHBOARD selects the user tab
    - Here is presented a table of users with all available information
    - For each user, there are 2 options, delete or update user
    - Admin selects delete user
    - A confirmation message is displayed
    - The admin returns to the DASHBOARD with the User and any trace removed
2) Admin updates an user
    - On admin DASHBOARD on User Tab, Administrator selects the Update User
    - The procedure and exceptions are the same when a user updates a profile
    - Returns to the dashboard with updated details
3) Admin deletes an event
    - On admin DASHBOARD the administrator selects the events tab

- Here is presented with a table of events each having 2 options, delete or update
- Administrator clicks on delete event
- A confirmation message is displayed
- Returns to the admin Page with event deleted

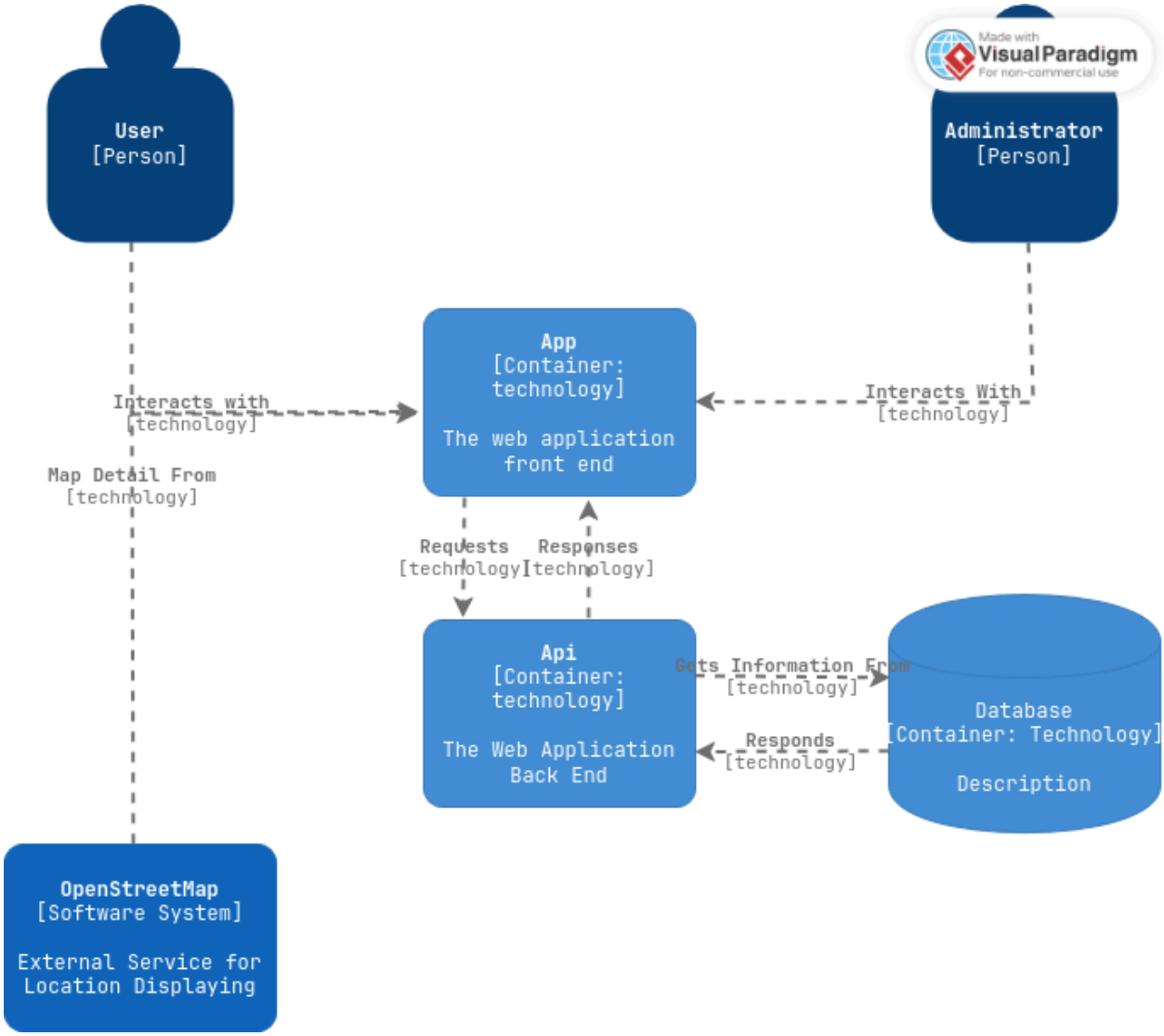
4) Admin Updates an event
   - On admin DASHBOARD at Event tab, Administrator selects update event
   - The procedure and exception is the same as a user updates his own event
   - Returns to the Event tab with modified event
5) Admin exports data into CSV file
   - On other options tab, the Admin selects export as CSV
   - The file is saved on User’s hard drive
   - ALTERNATIVELY: He can choose JSON instead of CSV
6) Admin imports data from CSV file
   - On other options tab, the Admin selects Import CSV
   - Here is prompted to provide the path to the CSV or supported file
   - Once loaded, everything reflects on the contents of the file
   - ALTERNATIVELY: He can choose JSON instead of CSV
   - EXCEPTION: Invalid file type -> An error will be displayed accordingly
   - EXCEPTION: CSV or JSON file contains invalid data -> An error will be displayed accordingly
   - EXCEPTION: CSV or JSON data already present in database -> data already exist will be displayed

C4 Diagram (First 3 layers)

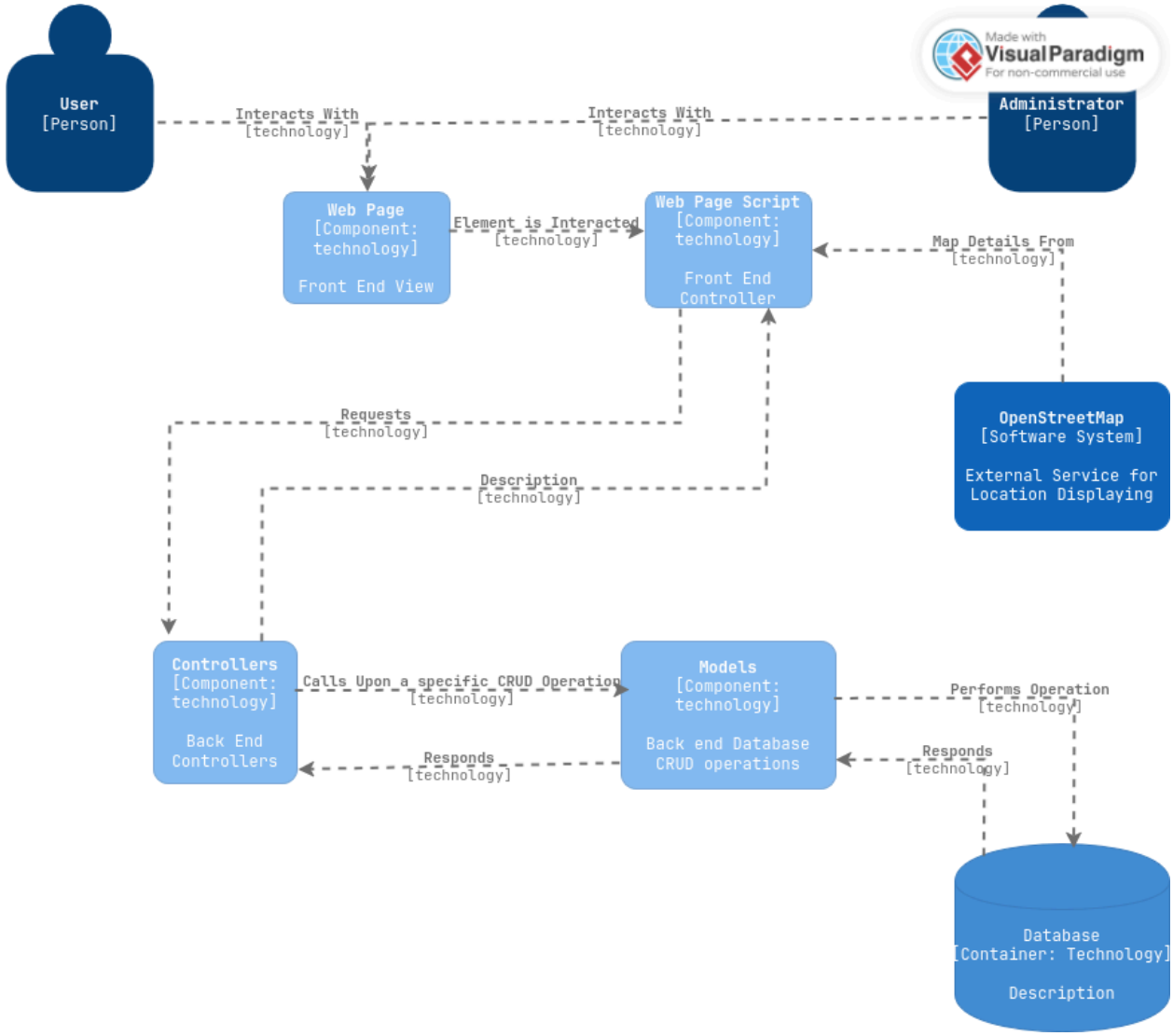
Context



Containers


User
[Person]
Administrator
[Person]
Made with
Visual Paradigm
For non-commercial use
App
[Container:
technology]
The web application
front end
Interacts with
[technology]
Interacts With
[technology]
Map Detail From
[technology]
Requests
[technology]
Responses
[technology]
Api
[Container:
technology]
The Web Application
Back End
Gets Information From
[technology]
Responds
[technology]
Database
[Container: Technology]
Description
OpenStreetMap
[Software System]
External Service for
Location Displaying

Component


User
[Person]
Interacts With
[technology]
Interacts With
[technology]
Administrator
[Person]
Web Page
[Component: technology]
Front End View
Element is Interacted
[technology]
Web Page Script
[Component: technology]
Front End Controller
Map Details From
[technology]
OpenStreetMap
[Software System]
External Service for Location Displaying
Requests
[technology]
Description
[technology]
Controllers
[Component: technology]
Back End Controllers
Calls Upon a specific CRUD Operation
[technology]
Models
[Component: technology]
Back end Database CRUD operations
Performs Operation
[technology]
Responds
[technology]
Responds
[technology]
Database
[Container: Technology]
Description

Project Structure

/api -> The backend component of the project
/api/admin -> Special Directory for admin space (Does not follow the MVC convention)
/api/config -> Directory for storing configurations, IE: Environments, JWT configuration etc.
/api/controllers -> Backend Controllers
/api/models -> Backend Models
/api/service -> Backend services (1 that is the Email service that doesn't work)

/app -> Frontend Component of the project
/app/helpers -> Helper functions
/app/templates -> Header and Footer of the pages
/app/pages -> Web applications views

/public -> Public folder
/public/css -> Stylesheets
/public/js -> JavaScript files (The front end controllers, responsible to send requests and receive responses from apis)
/public/images -> Example images for events and fields

Patterns used in developing this project:

- MVC Pattern: Program flow is separated in 3 categories:
    - Controllers: Handle incoming requests and orchestrates the application flow: Elements from /api/controllers
    - Models: that handle database interaction and data logic: Elements from /api/models
    - Views: Handles what is displayed to the user: /app/pages

- High Coupling: Controllers and pages often directly create instances of models, services, or helpers making them tightly bound to specific implementation

Why?

- Reusability: Code that is used to update the user and event details is the same for logged user and administrator
- Scalability: Implementation of further functionality is easier
- Separation of Concerns: Each file respects it's own functionality

- Layered Separation: Organized in multiple directories

Why?

- For better understanding which file should go

- Dependency Injection: The dependency for each controller is specified in the constructor.

  Auth Controller:

```
public function __construct($db)
{
        $this->userModel = new UserModel($db);
}
```

  User Model:

```
public function __construct($db)
{
        $this->db = $db;
}
```

- RESTful APIs: The controller receives HTTP requests and returns a response in JSON. In any worst case scenario, any error is handled with an accordingly HTTP code

Why?

- Allows scalability
- Allows Cacheability
- It follows client server architecture
- Allows layered system

- Allows stateless transfer of requests responses

Some HTTP Requests of the application

POST requests

- /api/index.php?action=register -> Given username, email and password as JSON, checks weather the same user with same username and email exists in the database and adds a new user entry to it
- /api/index.php?action=login -> Given email and password as JSON, checks weather the same user with same email exists in database
- /api/index.php?action=joinEvent -> Given user_id from JWT token and event_id as json, will add a new entry to EventParticipants table with the respective user and event
- /api/index.php?action=createEvent -> Given user_id from JWT token, and event details in JSON will create a new event with respective organizer Id
- /api/index.php?action=updateEvent -> Given user_id from JWT token and event details in JSON, it updates the event detail accordingly
- /api/index.php?action=deleteEvent -> Given the event_id as JSON, it deletes the respective event from the table
- /api/index.php?action=leaveEvent -> Given the user_id and event id it removes from EventParticipants the entry with the same user and event id given
- /api/index.php?action=sendRssFeed -> After the maximum participants is reached, given by the event id, will generate an rss feed consisting of event detail and a list of participants

GET requests

- /api/index.php?action=getEventRss -> Gets the generated Rss Feed
- /api/index.php?action=getEvents -> Returns all events from the table
- /api/index.php?action=getPastEvents -> Returns events that are over

PUT requests

- /api/index.php?action=updateProfile -> Given the user Id from JWT token and user details as JSON, it updates the user entry accordingly

Webgraphy

- MVC Architecture: https://www.geeksforgeeks.org/system-design/mvc-architecture-system-design/
- REST API: https://www.geeksforgeeks.org/node-js/rest-api-introduction/ https://restfulapi.net/
- JWT Token: https://www.sitepoint.com/php-authorization-jwt-json-web-tokens/
- Dynamic RSS generation: https://moldstud.com/articles/p-generating-dynamic-rss-feeds-with-php-a-comprehensive-developers-guide
- Website that I created the C4 Diagram: https://www.visual-paradigm.com/