universidade de aveiro
theoria poiesis praxis

EVENT FINDER

EGS PROJECT

MECT

- Bruno Silva – 97931
- Diogo Torrinhas - 98440
- João Torrinhas - 98435
- Miguel Tavares - 98448
- Tiago Bastos - 97590

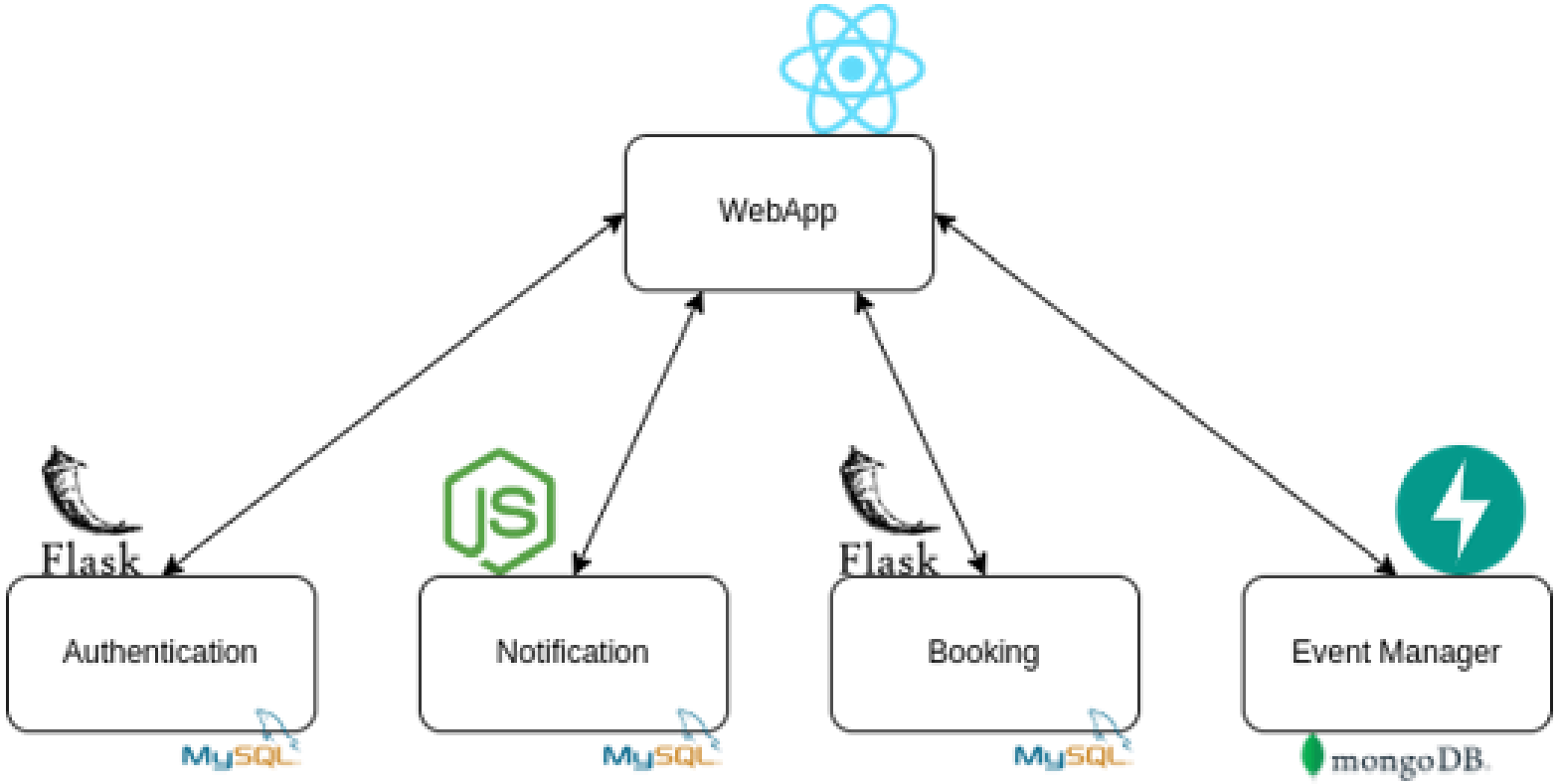
PROJECT IDEA

The idea of the project is to create an event management application that allows the search and purchase of tickets for some events or re-sell them.

This will consist of the following services:

- Authentication Service
- Booking Service
- Event Manager Service
- WebApp EventFinder Service
- Notification Service

ARCHITECTURE



SERVICES

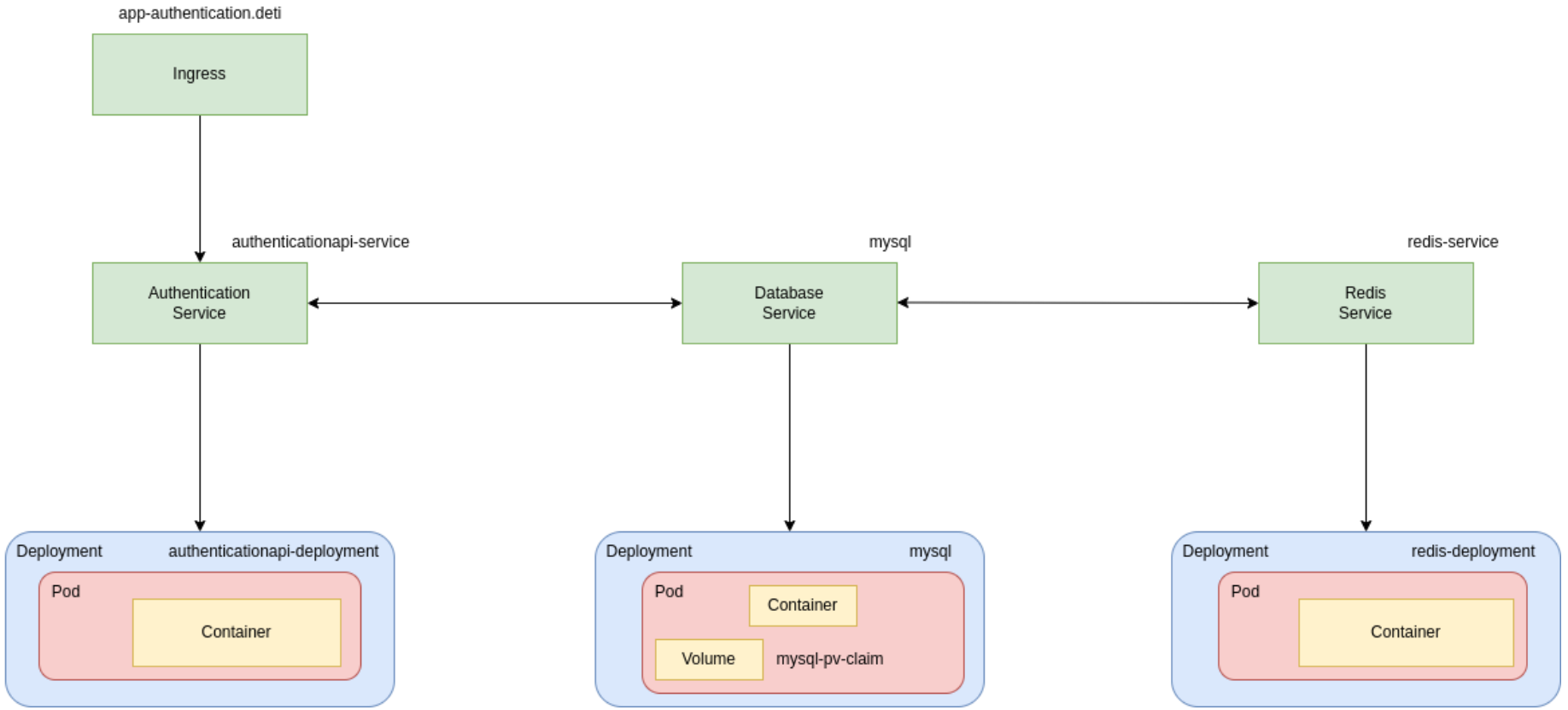
- **Authentication Service** - João Torrinhas
  - Manage system users (Add, Remove), user authentication and your account data
  - Validate user credentials
- **Event Manager Service** - Bruno Silva
  - Manage concerts available in the system (Add, Remove, Update, Delete and Search)
  - Events stored in a database
- **Booking Service –** Miguel Tavares
  - Manage the purchase of tickets for a specific event;
  - Pay for tickets easily and securely using Stripe;
  - All tickets purchased by a user are available on their page.
- **Notifications Service** - Diogo Torrinhas
  - Manage the creation and sending of notifications to users
  - The emails are managed by SendGrid
- **EventFinderService (Webapp)** - Tiago Bastos
  - Create a web application that aggregates and combines all services, in order to allow users to view available events and buy the respective tickets (if they wish) or re-sell their tickets

AUTHENTICATION API

API for user registration/login and validation.

- **POST /register** - Endpoint used to register a user.
- **POST /login** - Endpoint that checks a user's credentials and, if they are correct, a jwt token (access token) associated with it is returned.
- **POST /validate/{token_email}"** - Endpoint used to validate the email to complete the registration.
- **GET /verifyToken** - Endpoint used to check if the user is logged in through the token that was created when the user logged in.
- **DELETE /logout** - Endpoint used to log out a user. The user token is removed and added to the redis database (has the removed tokens).
- **GET /user/email/{email}** - Given an email, it returns the user id associated with it.

AUTHENTICATION DEPLOYMENT

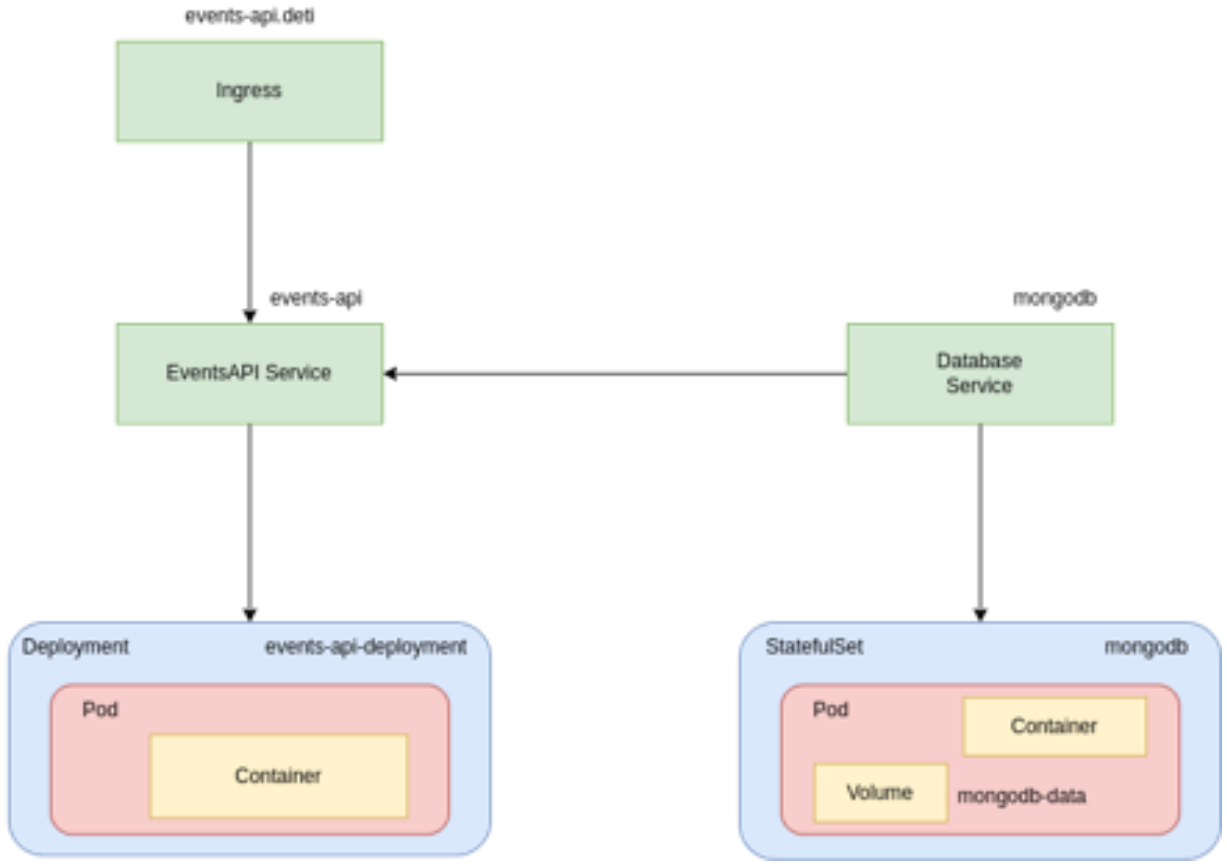


EVENT MANAGER API

API for event management:

- **GET /events -** Retrieve all the events
- **POST /events/** - Creates a new event
- **GET /events/{event_id}** - Get the event with the matching event id
- **PUT /events/{event_id}** - Updates the event with the given event id
- **DELETE /events/{event_id}** - Deletes the event with the given event id

EVENT MANAGER DEPLOYMENT

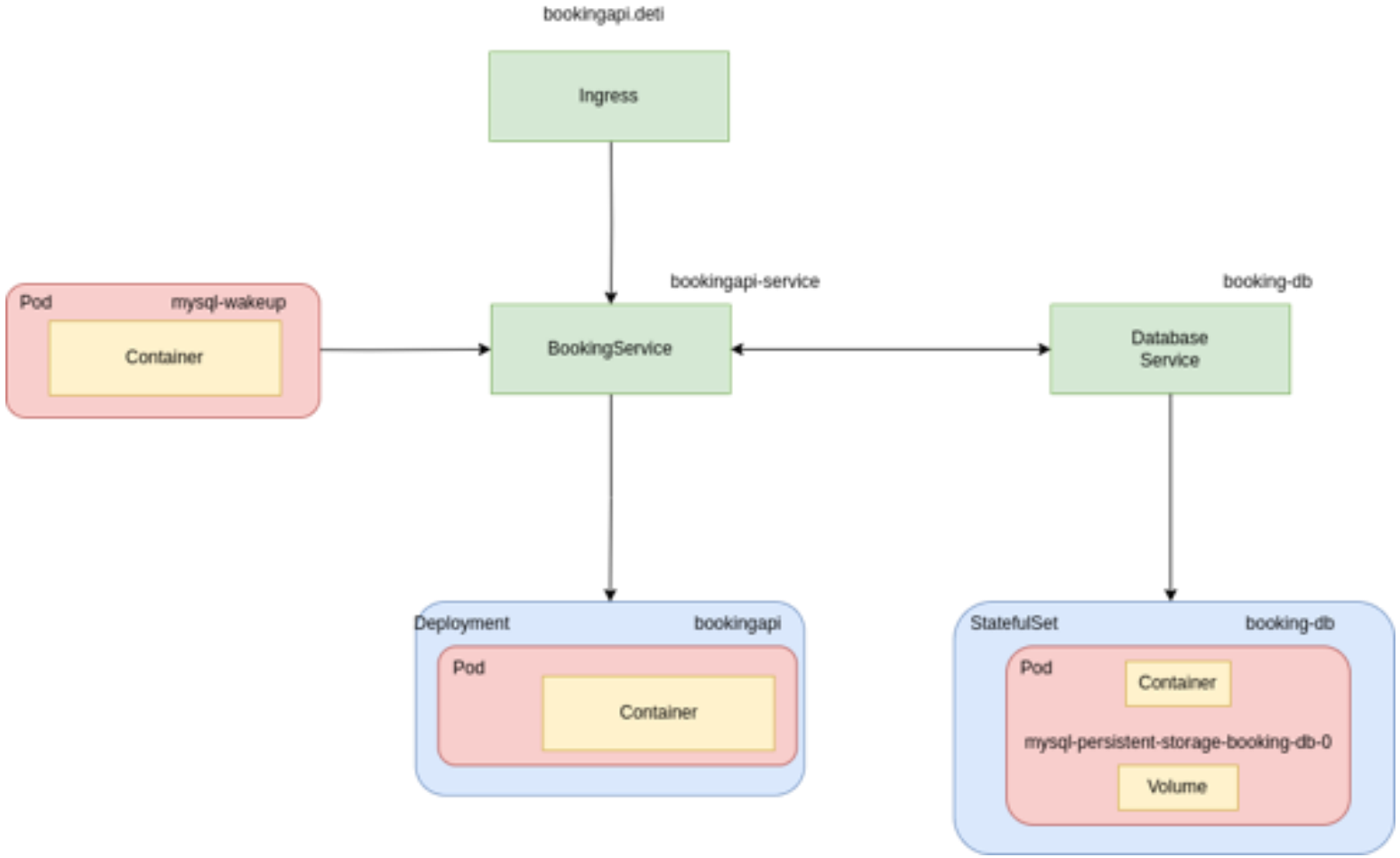


BOOKING API

API for booking tickets:

- **POST /ticket** - Book a ticket
- **DELETE /ticket/{ticket_id}** - Unbook a ticket
- **GET /ticket/cancel** - Cancel payment of the ticket
- **GET /ticket/success** - Checks if the payment was successful
- **POST /ticket/success** - Assigns the ticket to the user who paid for it
- **GET /ticket/{ticket_id}** - Get a booked ticket
- **GET /ticket/user/tickets** - Get all booked tickets for a user
- **DELETE /ticket/event/{event_id}** - Delete tickets given the event
- **GET /ticket/{ticket_id}/trade** - Ticket trade between two users
- **PUT /ticket/{ticket_id}/complete_trade/{token}** - Complete the trade of a ticket

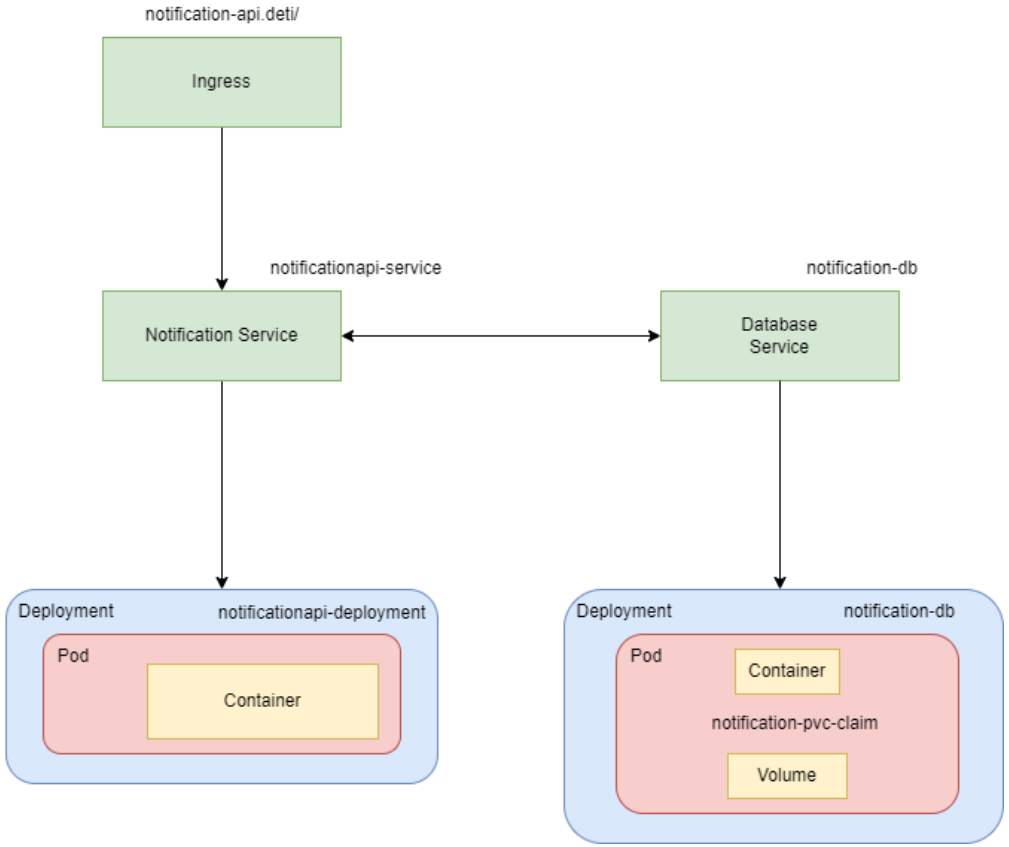
BOOKING DEPLOYMENT



NOTIFICATION API

- **POST /notification**- Send a notification to email
- **GET/notification** - Retrieve all the notifications
- **GET /notification/{email}**- Retrieve all notifications of a user
- **DELETE/notification/delete/email/{email}**- Delete all notifications of a user by email
- **DELETE/notification/delete/id/{id}**- Delete a notification by ID
- **POST/group/** - Create a group
- **GET/group/** - Get all users in each group
- **GET/group/{name}** - Retrieve groupID by name
- **DELETE/group/{id}**- Deletes a group by ID
- **PUT/group/{id}** - Add members to a group
- **POST/groupnotification** - Send notification email to a group

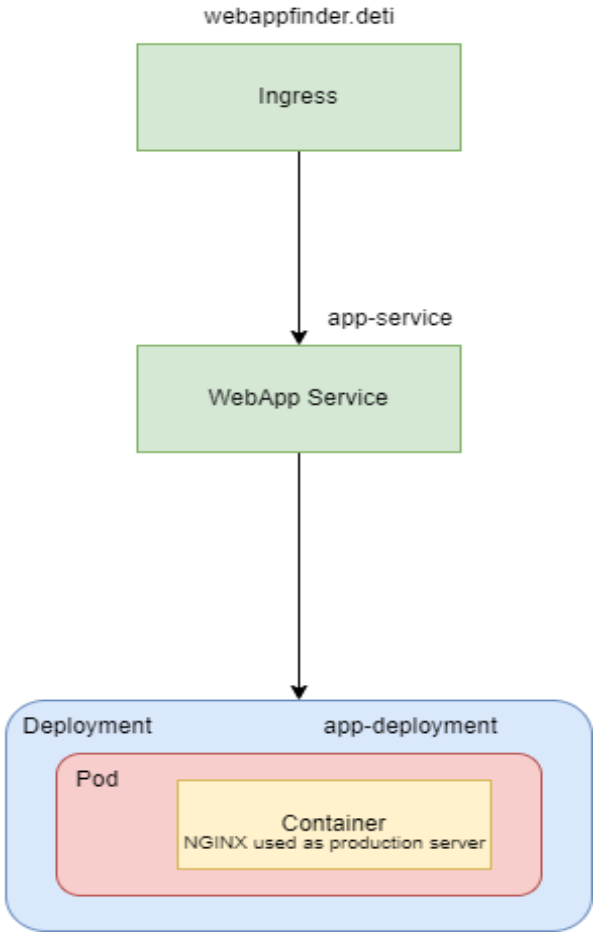
NOTIFICATION DEPLOYMENT



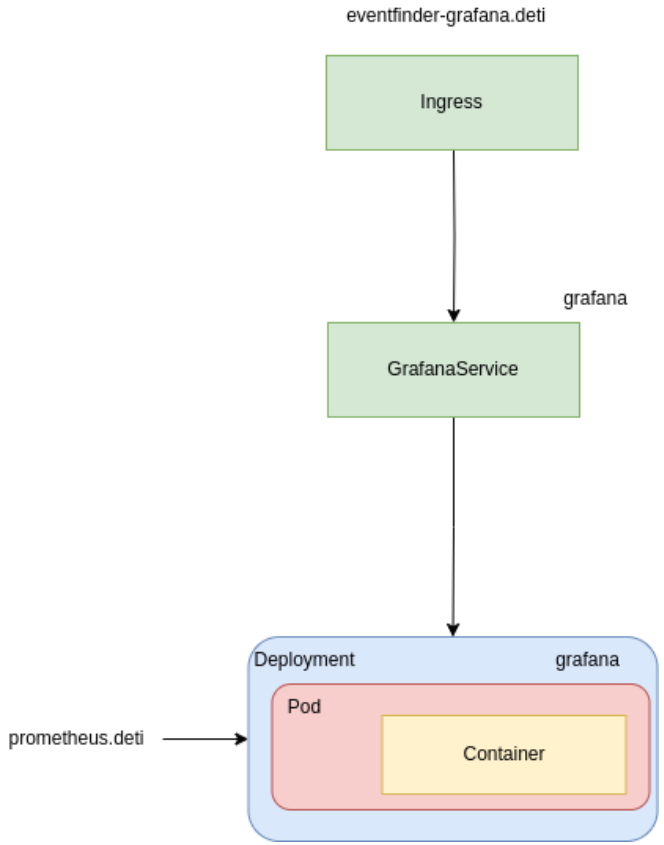
FRONT-END

- **Front-End Pages/Features:**
  - **Home Page**
  - **Events –** Page to display the Events Available.
  - **Tickets –** Page to display the tickets available for a selected Event.
  - **My Tickets -** Page to display the tickets of the User.
  - **Notifications –** Page to display the notifications of the User.
  - **Admin Page –** Page for the admin to manage (create, update and delete) Events.
  - **Login/Register -** Pages to Register/Create an Account and Login an User.

WEBAPP DEPLOYMENT



GRAFANA DEPLOYMENT



DEMONSTRATION

FUTURE WORK

Add new payment methods

Implement the customer support chat.

New authentication methods

CONCLUSION

- This project was very important in our academic path, as we got to know better the concept of services and how to interconnect them.
- In summary, we believe that we have effectively achieved the project's objectives, and overall, we have successfully met the goals of the course.
- Any questions ?
- Link for repositories and API's documentation:
  - https://github.com/orgs/EgsEventFinder/repositories
  - https://app.swaggerhub.com/apis-docs/EventFinder/BookingAPI/3.0.0
  - https://app.swaggerhub.com/apis-docs/DATORRINHAS44/Notifications/1.0.1
  - https://events-api.deti/docs
  - https://app.swaggerhub.com/apis-docs/EventFinder_Auth/ConcertFinder/1.0.0

AUTHORS CONTRIBUTION

- Bruno Silva – 20%
- Diogo Torrinhas - 20%
- João Torrinhas - 20%
- Miguel Tavares - 20%
- Tiago Bastos - 20%