Rengars App - dokumentace

Tým:

David **Bulko** (bulkodav), Václav **Smítka** (smitkva1), David **Stražovan** (strazdav), Petr **Uhlíř** (uhlirpe4)

Popis:

Aplikace Rengars by měla sloužit firmám ke snadnějšímu nalezení nových zaměstnanců, stejně tak zájemcům k nalezení nové práce. Tento nástroj by měl být podobný portálům prace.cz, jobs.cz apod.

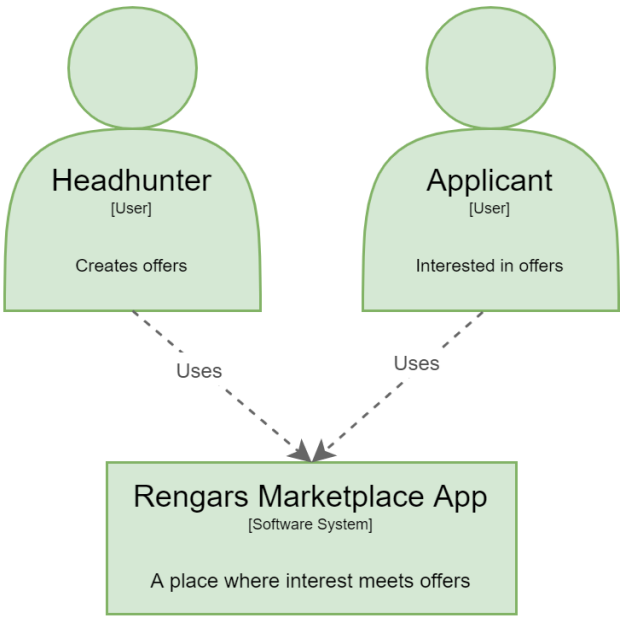
V rámci semestrální práce v kurzu Softwarové architektury byl navržen backend celé aplikace, konkrétně 4 microservisy, které obsluhují dané oblasti aplikace.

Aplikaci mohou využívat dva typy uživatelů - tzv. *Headhunteři*, tedy zástupci firem, kteří se snaží získat nové zaměstnance přidáváním pracovních nabídek, a tzv. *Applicanti*, tedy běžní uživatelé, zájemci o práci, kteří reagují na nabídky.

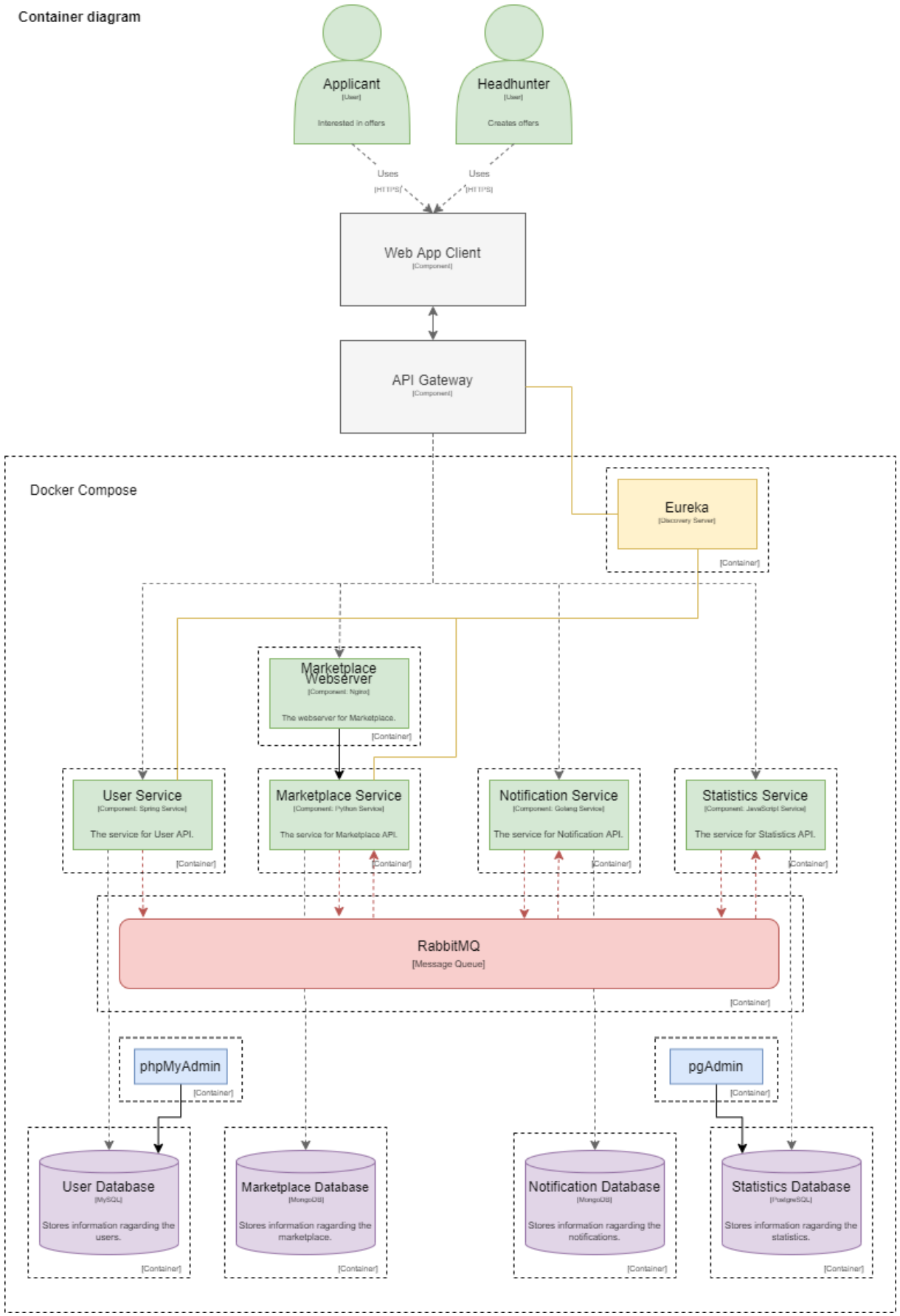
O správu uživatelů aplikace se stará *User service*. Středobodem aplikace je *Marketplace service*, která umožňuje headhunterům přidávat pracovní nabídky, na které budou dostávat odpovědi od applicantů. Aby se o nových nabídkách applicanti dozvěděli, budou dostávat oznámení, které zajišťuje *Notification service*. Dostanou notifikaci i o změně nabídky - popisu, stavu. Headhunteři naopak mohou dostat oznámení, že někdo zareagoval na jejich nabídku. Poslední službou je *Statistics service*, která provádí různé statistiky nad celou aplikací. Např. počet úspěšných nabídek headhuntera, počet zobrazení dané nabídky apod.

C4 MODEL:

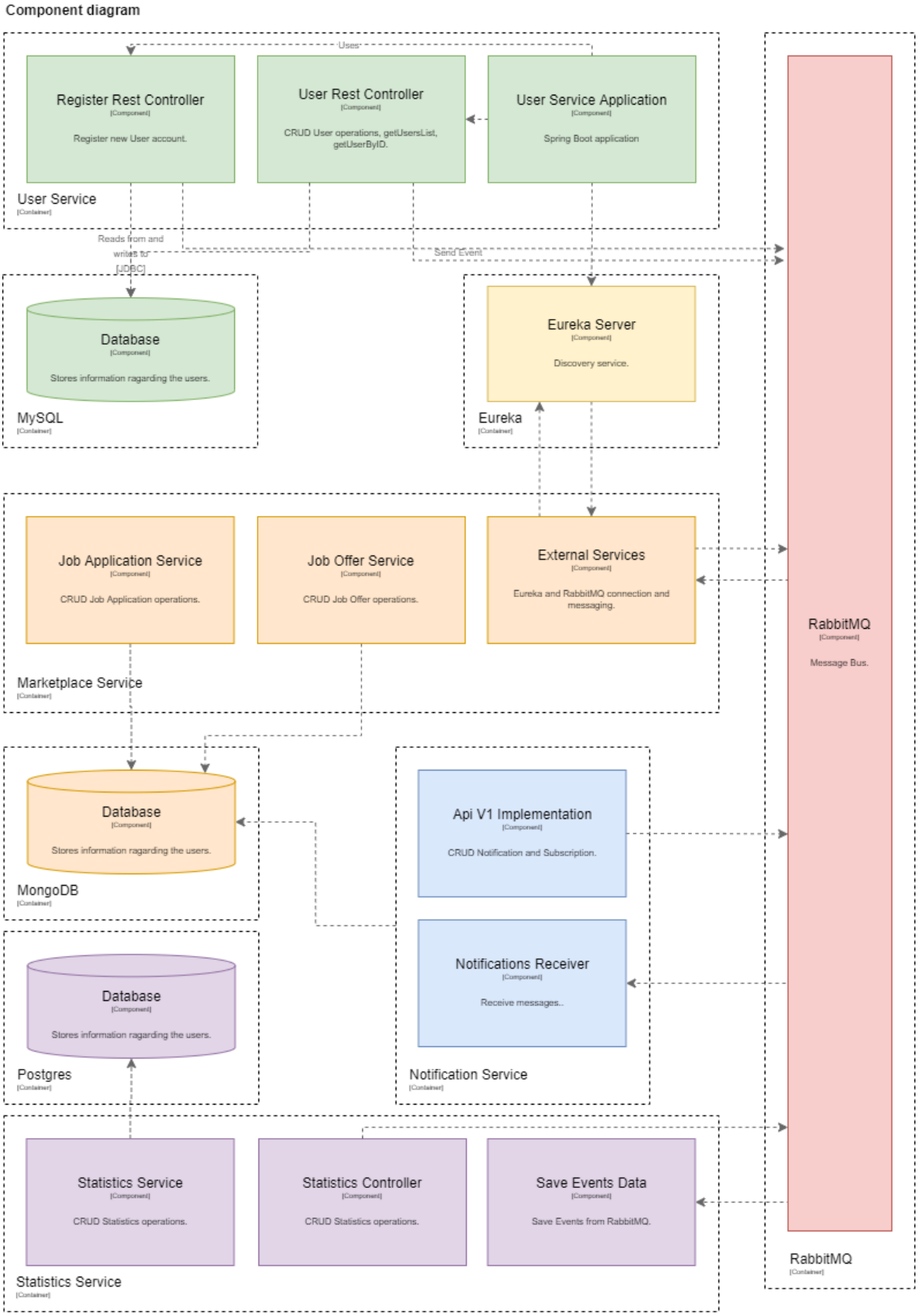
Context diagram



Container diagram



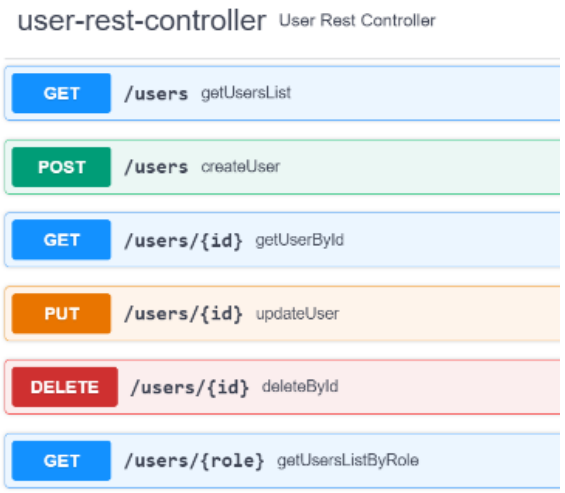
Component diagram



Code

Servisy:

User service



Funkce správy uživatelů:

- vytvoření nového uživatelského účtu
- načtení jednoho uživatelského účtu pomocí jeho id
- získání všech uživatelských účtů
- aktualizace údajů o uživatelském účtu
- smazání uživatelského účtu
- standardní ověření pro e-mail, telefonní číslo, heslo

DTOs:

- UserDTO
- UserListDTO
- ContactDTO
- AddressDTO
- RegisterUserAccountDTO
    - String username
    - String password
    - String name
    - String surname
    - String email
    - String gender
    - String role
- CreateOrUpdateUserDTO
    - String username
    - String password
    - String name
    - String surname
    - String gender

        - Date birthDate
        - Date createdAt
        - Date updateddAt
        - String email
        - String phoneNumber
        - String linkedin
        - String address
        - String address2
        - String city
        - String country
        - String zipCode
        - String role

Entities

- Address
- Contact
- Gender (enum)
    - MALE, FEMALE
- Role (enum)
    - APPLICANT, HEADHUNTER
- User

Rest API

- RegisterRestController
    - registerNewUserAccount(RegisterUserAccountDTO)
- UserRestController
    - getUsersList()
    - createUser(CreateOrUpdateUserDTO)
    - getUserById(Long)
    - updateUser(Long, CreateOrUpdateUserDTO)
    - deleteUser(Long)
    - getUsersListByRole(Long)

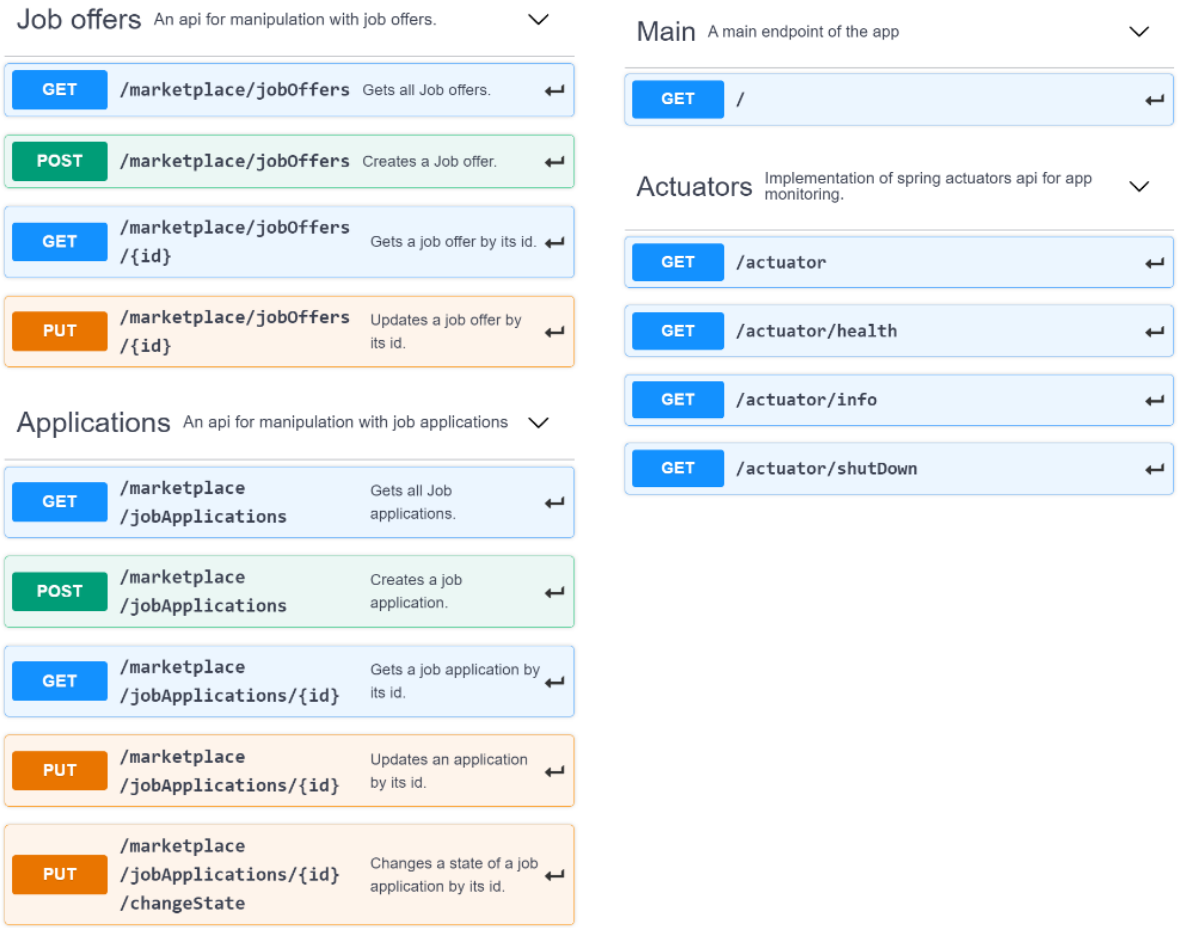
Classes

- UserServiceApplication
    - connectionFactory()
    - main()
    - jsonMessageConventer()
    - rabbitTemplate
- SpringBootUtil
- Events
    - createEvent()
- RabbitMQSender
    - send()
- UserService
    - getUsersList()
    - getUserById()
    - getUserByUsername()
    - getUserByEmail()

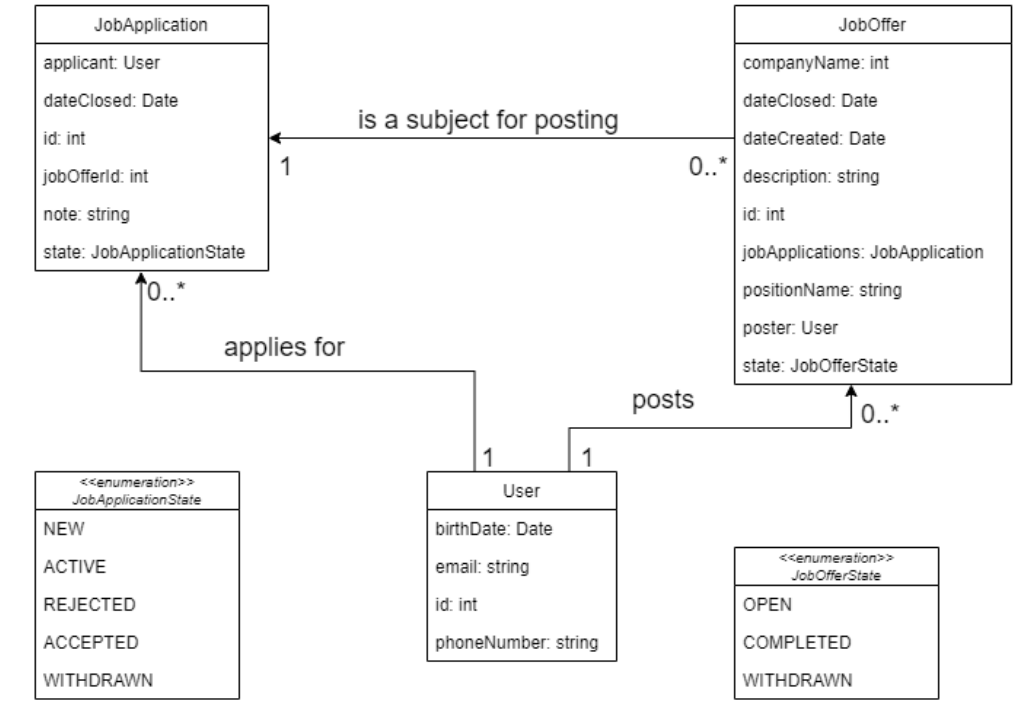
- registerUserAccount()
- checkIfUsernameNotUsed()
- checkIfEmailNotUsed()
- createuser()
- addContactOnUser()
- addAddressOnUser()
- updateUser()
- getUserList()
- deleteUserById()

Marketplace service

Funkce Marketplace service:



Marketplace
Class diagram



Notification service

Notifications

- **GET** `/v1/notifications` — Gets all notifications. By default filters the list to cointain only queued notifications.
- **POST** `/v1/notifications` — Creates new notification entry
- **GET** `/v1/notifications/{id}` — Gets a notification by its id
- **DELETE** `/v1/notifications/{id}` — Cancels notification with given id

Schemas

NotificationPostEntry {

type*	string *example: OFFER_CREATED* Type of the notification. Message template and configuration for sending the message is determined by this.
parameters	NotificationParameteres {...} *example: OrderedMap { "offerName": "SUPER FIRE OFFER BE A MILLIONARE IN A MONTH", "company": "SILLICON VALLEY CRYPTO BARONS" }*
sendAt	string *example: 2020-04-18T22:00:00Z* An UTC datetime when the message should be send. Be aware that the notification won't be necessarily send at the specified time.

}

NotificationQueueEntry {

id*	string *example: AAAA-BBBB-1233-CDEFG* An ID of the created notification. Can be used in other calls.
status*	NotificationStatus string State of the notification Enum: [QUEUED, CANCELLED, SENT]
parameters	NotificationParameteres {...} *example: OrderedMap { "offerName": "SUPER FIRE OFFER BE A MILLIONARE IN A MONTH", "company": "SILLICON VALLEY CRYPTO BARONS" }*
scheduledAt*	string *example: 2020-04-18T22:00:00Z* An UTC datetime at which the notification will be send. Be aware that the notification won't be necessarily send exactly at this time.
sentAt	string *example: 2020-04-18T22:00:00Z* An UTC datetime when the notification was sent. Present only for notifications in SENT state.

}

NotificationParameteres {

description:	Object containing parameters for the message template

}
example: OrderedMap { "offerName": "SUPER FIRE OFFER BE A MILLIONARE IN A MONTH", "company": "SILLICON VALLEY CRYPTO BARONS" }

NotificationStatus string

State of the notification

Enum:

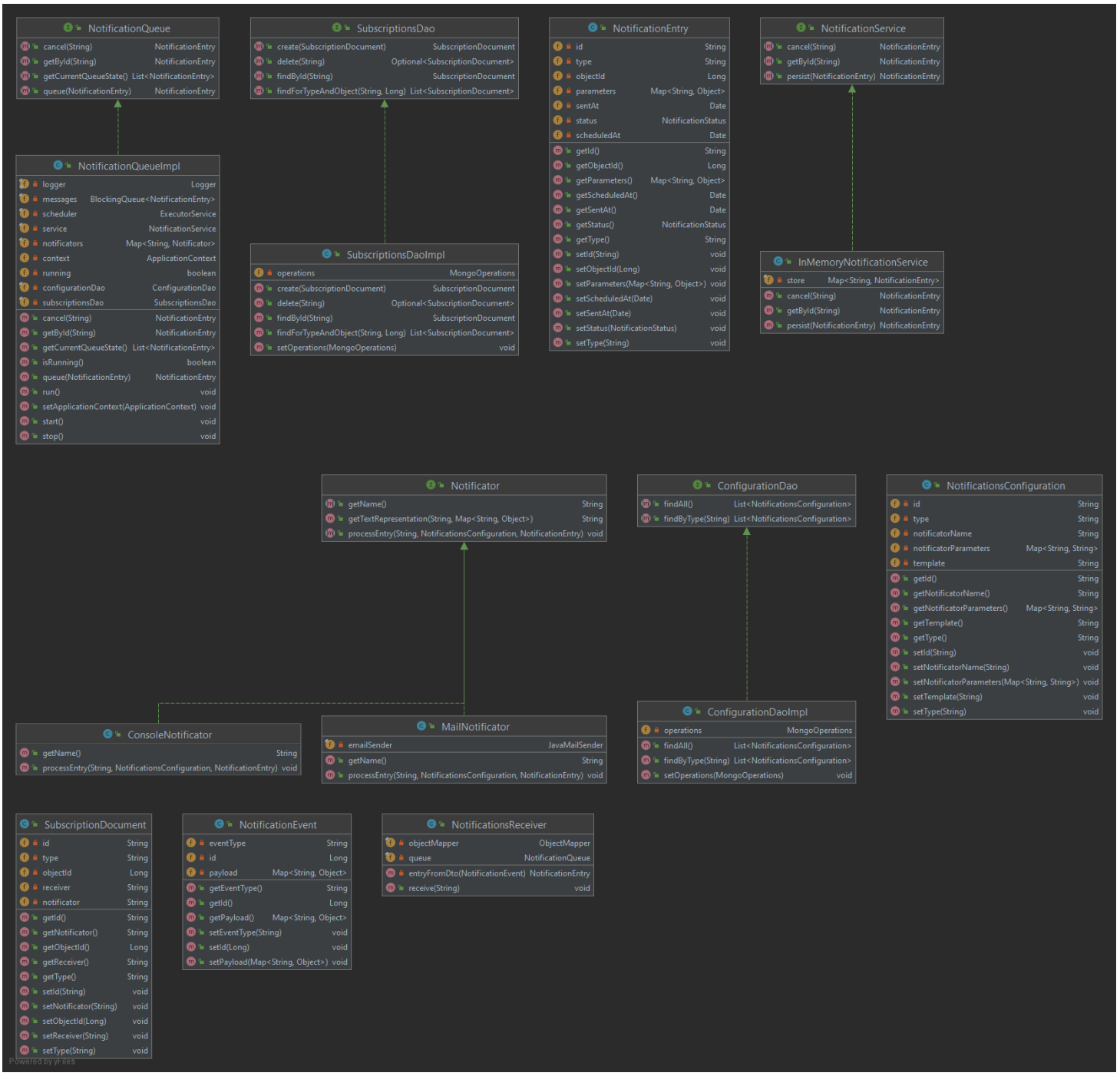
[QUEUED, CANCELLED, SENT]

Error {

description:	Describes error, its identifier and additional info.
type*	string *example: errors.notifications.alreadySent* Unique id of the error
message	string *example: Sent notification can not be cancelled.* Error message
details	{...}

}

Class diagram:



Statistics service

Events
Entry of all events.

POST `/events/{eventType}` Add a new event to the store

Statistics
Everything about statistics.

GET `/basicStatistisc` Gets all basic statistics.

GET `/statistics/newOffers` Gets new offers in time.

GET `/statistics/newOffers/headhunter/{id}` Gets new offers in time for specific headhunter.

UserEvent {

id	integer($int32)
firstName	string
lastName	string
birthDate	string($date)
createdAt	string($date-time)
eventType	UserEventType string Enum: [NEW_HEADHUNTER, UPDATE_HEADHUNTER, DELETE_HEADHUNTER, NEW_APPLICANT, UPDATE_APPLICANT, DELETE_APPLICANT]

}

OfferEvent {

offerId	integer($int32)
userId	integer($int32)
companyName	string
positionName	string
description	string
createdAt	string($date-time)
eventType	OfferEventType string Enum: [NEW_OFFER, UPDATE_OFFER, SUCCESS_CLOSE_OFFER, FAILURE_CLOSE_OFFER]

}

NewOffers {

offers	[{ offerId integer($int32) headhunter integer($int32) companyName string positionName string createdAt string($date-time) }]

}

Schemas

EventType string
Enum:
[user, offer, application]

UserEventType string
Enum:
[NEW_HEADHUNTER, UPDATE_HEADHUNTER, DELETE_HEADHUNTER, NEW_APPLICANT, UPDATE_APPLICANT, DELETE_APPLICANT]

OfferEventType string
Enum:
[NEW_OFFER, UPDATE_OFFER, SUCCESS_CLOSE_OFFER, FAILURE_CLOSE_OFFER]

ApplicationEventType string
Enum:
[NEW_APPLICATION, UPDATE_APPLICATION, SUCCESS_CLOSE_APPLICATION, FAILURE_CLOSE_APPLICATION]

ApplicationEvent {

applicationId	integer($int32)
userId	integer($int32)
note	string
jobOfferId	integer
createdAt	string($date-time)
eventType	ApplicationEventType string Enum: [NEW_APPLICATION, UPDATE_APPLICATION, SUCCESS_CLOSE_APPLICATION, FAILURE_CLOSE_APPLICATION]

}

BasicStatistics {

numberOfHeadhusters	integer($int32)
numberOfUsers	integer($int32)
numberOfOffers	integer($int32)
numberOfSuccessCloseOffers	integer($int32)
numberOfFailureCloseOffers	integer($int32)
ratioOfSuccessfulToUnsuccessfulOffers	number($double)
ratioOfHeadhunresToUsers	number($double)

}

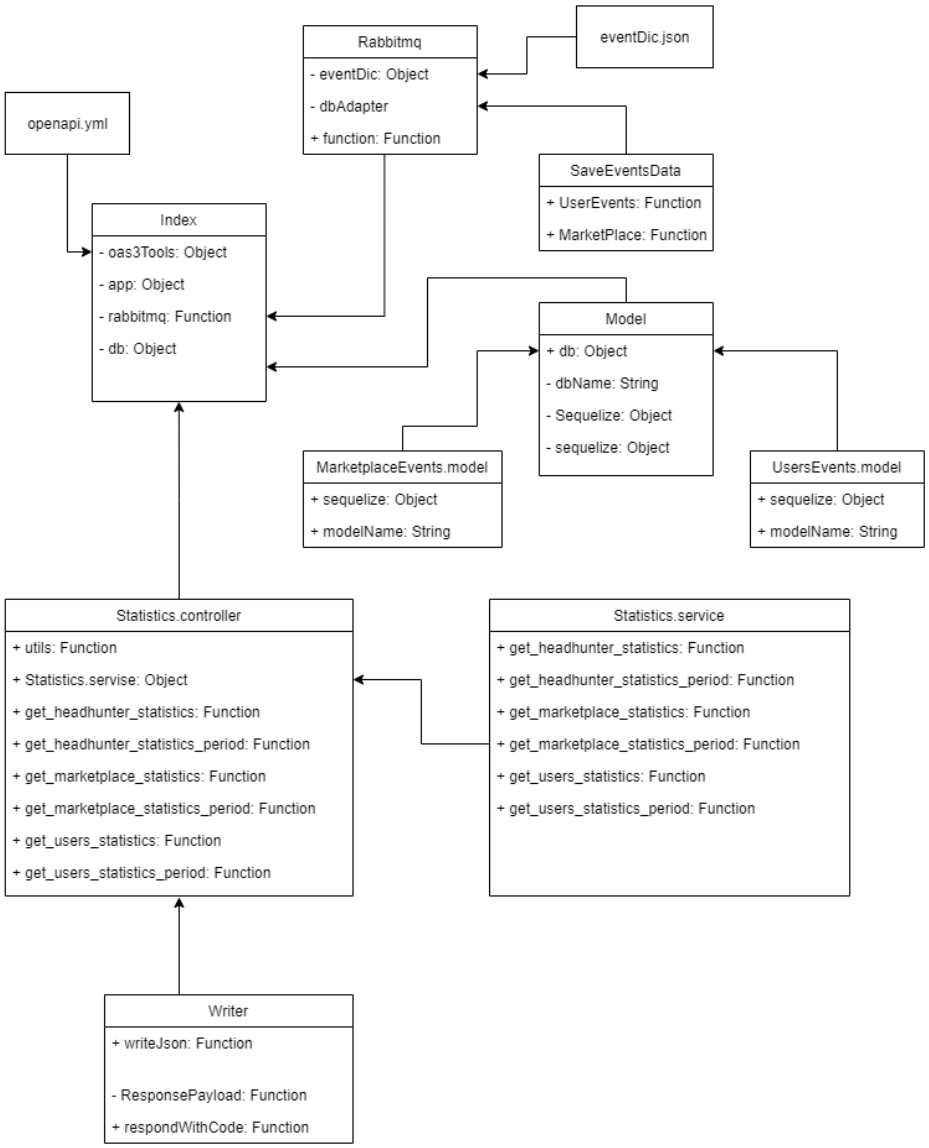
NewOffersInTime {

```
headhunter      integer($int32)
offers          [ {
                  offerId       integer($int32)
                  companyName   string
                  positionName  string
                  createdAt     string($date-time)

                }]
}
```

Error {

```
description:
                Describes error, its identifier
                and additional info.

type*           string
                example:
                errors.notifications.alreadySent

                Unique id of the error

message         string
                example: Sent notification can
                not be cancelled.

                Error message

details         > {...}
}
```

Class/component diagram
Statistics Service



Rengars App repository:

https://github.com/VaSmitka/SWA_Rengars