Нотификации-Финтех

# Описание

Веб-приложение, обладающее следующим функционалом: отправка и составление запросов к REST API, вывод ответа на запрос на страницу в качестве Json файла.

## Наименование

Нотификации-Финтех

## Предметная область

Разработка прикладного ПО

# Данные

В приложении будут использоваться следующие модели данных:

Project – проект в котором работает сотрудник. Проект включает в себя название и описание.

CREATE TABLE project  
(  
 id SERIAL PRIMARY KEY,  
 name VARCHAR(50) NOT NULL,  
 description TEXT  
);

Employee – работник компании, который работает в каких то проектах. Работник включает в себя телефон, имя, дату рождения, дату начала работы, город и его логин в системе, который ссылается на таблицу для работы со Spring Security. При удалении юзера из таблицы информация о работнике тоже удалится.

CREATE TABLE employee  
(  
 id SERIAL PRIMARY KEY,  
 phone TEXT,  
 full\_name VARCHAR(100) NOT NULL,  
 birthday DATE NOT NULL,  
 start\_work DATE NOT NULL,  
 city VARCHAR(50) NOT NULL

username VARCHAR(255);

CONSTRAINT fk\_username FOREIGN KEY (username) REFERENCES users (username) ON DELETE CASCADE;

);

employee\_project\_relationship – таблица для связи ManyToMany работника и проекта.

CREATE TABLE employee\_project\_relationship  
(  
 employee\_id BIGINT,  
 project\_id BIGINT,  
 FOREIGN KEY (employee\_id) REFERENCES employee (id) ON DELETE CASCADE,  
 FOREIGN KEY (project\_id) REFERENCES project (id) ON DELETE CASCADE,  
 UNIQUE (employee\_id, project\_id)  
);

Present – подарок, который желает работник на свой праздник. Present содержит в себе название, описание, ссылку на сайт, где его можно купить. Подарок должен принадлежать какому-то работнику, поэтому есть ссылка на работника из employee.

CREATE TABLE present  
(  
 id SERIAL PRIMARY KEY,  
 name VARCHAR(50) NOT NULL,  
 description TEXT,  
 link VARCHAR(255),  
 employee\_id BIGINT NOT NULL,  
 FOREIGN KEY (employee\_id) REFERENCES employee (id) ON DELETE CASCADE  
);

Notification – уведомление о празднике работника. Содержит в себе тип праздника (либо день рождения, либо годовщина работы), ссылку на работника, у которого этот праздник. Данные генерируются автоматически раз день.

CREATE TABLE notification  
(  
 id SERIAL PRIMARY KEY,  
 type VARCHAR(15) NOT NULL,  
 employee\_id BIGINT NOT NULL,  
 FOREIGN KEY (employee\_id) REFERENCES employee (id) ON DELETE CASCADE,  
 CONSTRAINT valid\_type CHECK (type = 'BIRTHDAY' OR type = 'ANNIVERSARY')

CONSTRAINT unique\_type\_employee UNIQUE (type, employee\_id);

);

Colleague\_notification – уведомление для коллег. Хранит ссылки на нотификацию и на работника, которому отправится эта нотификация.

CREATE TABLE colleague\_notification  
(  
 id SERIAL PRIMARY KEY,  
 notification\_id BIGINT,  
 colleague\_id BIGINT,  
 FOREIGN KEY (notification\_id) REFERENCES notification (id) ON DELETE CASCADE,  
 FOREIGN KEY (colleague\_id) REFERENCES employee (id) ON DELETE CASCADE,  
 CONSTRAINT unique\_notification\_colleague UNIQUE (notification\_id, colleague\_id)  
);

User – пользователь приложения. Модель нужна для авторизации работника, который заходит в аккаунт. Содержит в себе логин, пароль и значение активен ли пользователь.

create table users  
(  
 username varchar(50) not null primary key,  
 password varchar(100) not null,  
 enabled boolean not null  
);

Authority – право пользователя. Содержит ссылку на пользователя, которому относится право и само название права.

create table authorities  
(  
 username varchar(50) not null,  
 authority varchar(50) not null,  
 constraint fk\_authorities\_users foreign key (username) references users (username)  
);

# Пользовательские роли

Существует две роли: user и admin

User:

* Может смотреть данные других работник (их город, день рождения, желаемые подарки, проекты)
* Может смотреть свои уведомления о праздниках
* Не может смотреть чужие уведомления
* Не может менять данные чужого аккаунта
* Не может менять данные подарков чужого аккаунта
* Не может создавать нотификации
* Могут добавлять себе новый проект

Admin:

* Может изменять данные любого аккаунта
* Может изменять и создавать чужие подарки
* Может вручную создавать уведомления о праздниках и изменять их
* Может смотреть любую информацию о пользователях
* Может добавлять пользователям проекты и создавать новые проекты

API приложения  
В данном проекте реализован следующий API:

* API метод, возвращающий JSON с данными работника.

Путь:/api/employee/get

Аргументы employee\_id – id работника.

Пример запроса /api/employee/get?employee\_id=1.

Структура ответа: данные работника с указанным id

{

"id": 12,

"full\_name": "Zubenko Mikhail Petrovich11",

"phone": "+79992434271",

"projects": [],

"birthday": "2002-12-28",

"start\_work": "2018-10-10",

"wishlist": [

{

"id": 19,

"name": "Present to 11",

"link": "https://github.com",

"description": "Description4",

"employee\_id": 12

}

],

"city": "London",

"username": "user16"

}

* API метод, возвращающий JSON с данными нотификации.

Путь:/api/notification/get

Аргументы notification\_id – id нотификации.

Пример запроса /api/notification/get?notification\_id=1.

Структура ответа: данные нотификации с указанным id

{

"id": 1,

"type": "BIRTHDAY",

"employee": {

"id": 2,

"full\_name": "Zubenko Mikhail Petrovich2",

"phone": "+79992434271",

"projects": [

{

"id": 1,

"name": "Just Tinkoff1",

"description": "Just description1"

}

],

"birthday": "2002-02-10",

"start\_work": "2018-10-10",

"wishlist": [

{

"id": 4,

"name": "Present4",

"link": "link4",

"description": "Description4",

"employee\_id": 2

},

{

"id": 5,

"name": "Present5",

"link": "link5",

"description": "Description5",

"employee\_id": 2

}

],

"city": "London",

"username": "user2"

}

}

* API метод, возвращающий JSON с данными подарка.

Путь:/api/present/get

Аргументы present\_id – id подарка.

Пример запроса /api/present/get?present\_id=1.

Структура ответа: данные подарка с указанным id и с работником которому подарок принадлежит.

{

"id": 19,

"name": "Present to 11",

"link": "https://github.com",

"description": "Description4",

"employee\_id": 12

}

* API метод, возвращающий JSON с данными проекта.

Путь:/api/project/get

Аргументы project\_id – id проекта.

Пример запроса /api/project/get?project\_id=1.

Структура ответа: данные проекта с указанным id

{

"id": 1,

"name": "Just Tinkoff1",

"description": "Just description1"

}

* API метод, регистрирующий пользователя.

Путь:/auth/register

Тело запроса: данные работника с логином и паролем.

Пример запроса /auth/register

Пример тела запроса:

{"full\_name": "Zubenko Mikhail Petrovich11",

"phone": "+79992434271",

"birthday": "2002-12-28",

"start\_work": "2018-10-10",

"city": "London",

"username": "user15",

"password": "user"}

Структура ответа: данные работника с сохраненным id (как в get employee)

* API метод, сохраняющий подарок и возвращающий JSON с данными подарка.

Путь:/api/present/save

Аргументы employee\_id – id работника, которому принадлежит этот подарок.

Пример запроса /api/present/save?employee\_id=1

Пример тела запроса:

{

"name": "Present to 1",

"link": "https://github.com",

"description": "Description4"

}

Структура ответа: данные подарка с сохраненным id (как в get present)

* API метод, сохраняющий проект и возвращающий JSON с данными проекта.

Путь:/api/project/save

Пример запроса /api/project/save?

{

"name": "Just Tinkoff1",

"description": "Just description1"

}

Структура ответа: данные проекта с сохраненным id (как в get project).

* API метод, обновляющий пользователя.

Путь:/api/employee/update

Тело запроса: обновленные данные работника и пароль с логином.

Пример запроса:/api/employee/update

Пример тела запроса:

{“id”: “1”,

"full\_name": "Zubenko Mikhail Petrovich11",

"phone": "+79992434271",

"birthday": "2002-12-28",

"start\_work": "2018-10-10",

"city": "London",

}

Структура ответа: код ответа

* API метод, обновляющий подарок.

Путь:/api/present/update

Тело запроса: обновленные данные подарка и пароль с логином

Пример запроса:/api/employee/update

Пример тела запроса:

{

"id": 4,

"name": "Present to 11",

"link": "https://github.com",

"description": "Description4",

}

Структура ответа: код ответа

* API метод, обновляющий проект.

Путь:/api/project/update

Тело запроса: обновленные данные проекта и пароль с логином

Пример запроса:/api/project/update

Пример тела запроса:

{

"id": 4,

"name": "Just Tinkoff1",

"description": "Just description1"

}

Структура ответа: код ответа

* API метод, добавляющий работника в проект.

Путь:/api/project/addEmployee

Аргументы project\_id – id проекта, employee\_id – id работника.

Пример запроса:/api/project/update

Структура ответа: код ответа

* API метод, удаляющий пользователя.

Путь:/api/employee/delete

Тело запроса: данные работника и пароль с логином.

Пример запроса:/api/employee/delete

Пример тела запроса:

{“id”: “1”,

"full\_name": "Zubenko Mikhail Petrovich11",

"phone": "+79992434271",

"birthday": "2002-12-28",

"start\_work": "2018-10-10",

"city": "London",

}

Структура ответа: код ответа

* API метод, удаляющий подарок.

Путь:/api/present/delete

Тело запроса: данные подарка и пароль с логином

Пример запроса:/api/present/delete

Пример тела запроса:

{

"id": 4,

"name": "Present to 11",

"link": "https://github.com",

"description": "Description4",

}

Структура ответа: код ответа

* API метод, удаляющий проект.

Путь:/api/project/delete

Тело запроса: данные проекта и пароль с логином

Пример запроса:/api/project/delete

Пример тела запроса:

{

"id": 4,

"name": "Just Tinkoff1",

"description": "Just description1"

}

Структура ответа: код ответа

* API метод, возвращающий JSON с нотификациями работника.

Путь:/api/employee/getMyNotifications

Аргументы employee\_id – id работника.

Пример запроса /api/employee/ getMyNotifications?employee\_id=1.

Структура ответа: выводятся все уведомления о праздниках коллег на неделю вперед, уведомление будет приходить, только если работники участвуют в одном проекте.

[

{

"id": 28,

"type": "ANNIVERSARY",

"employee": {

"id": 9,

"full\_name": "Zubenko Mikhail Petrovich9",

"phone": "+79992434271",

"projects": [

{

"id": 4,

"name": "Just Tinkoff4",

"description": "Just description4"

}

],

"birthday": "2002-10-28",

"start\_work": "2018-02-11",

"wishlist": [

{

"id": 14,

"name": "Present14",

"link": "link14",

"description": "Description14",

"employee\_id": 9

}

],

"city": "London",

"username": "user9"

}

}

]

## Язык программирования

В проекте использовались следующие языки программирования и технологии: Java 17, MyBatis для работы с бд и маппинга sql запросов, веб-фреймворк Spring boot, Flyway для контроля версий схемы бд, PostgreSQL, gitlab-ci для деплоя, Docker Testcontainers для разворачивания бд в тестах, Spring Security для авторизации и аутентификации, Hibernate-Validator для валидации входных данных. Архитектура приложения: клиент-сервер.

## СУБД

PostrgeSQL