МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ
ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»

Основная профессиональная образовательная программа
Направление подготовки 09.03.01 Информатика и вычислительная техника
Направленность (профиль) «Технологии разработки программного обеспечения»
форма обучения – очная

ОТЧЁТ

по реализации проекта для дисциплины «Базы данных»
по направлению “09.03.01 – Информатика и вычислительная техника”
(профиль: “Технологии разработки программного обеспечения ”)

Преподаватель: к.ф-м.н., доцент кафедры ИТиЭО

_____
(Жуков Н. Н.)

Студент 2 курса:

Мельников Ф. В. _____

Санкт-Петербург
2021

Оглавление

Ответственные 3
Предметная область 3
Ход выполнения нормализации 3
Объяснение выбранной СУБД 4
Описание сущностей 4
ER-диаграмма 7
Реализация REST API 8
Исходный текст запросов 9
Запрос по созданию таблиц 9
Запрос по созданию представления 15
Заключение 16

Ответственные

Мельников Ф. В. – разработчик проекта. В обязанности разработчика входили процессы определения отношений для заданной предметной области, нормализации полученных отношений, создания схемы базы данных, реализации программного проекта.

Предметная область

Региональный центр содействия трудоустройству и адаптации к рынку труда выпускников организует биржу вакансий. Вакансии поступают от организаций и предприятий. Студенты обращаются в Центр с целью поиска работы. Соискатели вакансий самостоятельно или через операторов составляют резюме, которые проверяются операторами и просматриваются работодателями. Специалисты Центра (операторы) подбирают подходящие вакансии.

Информационная система должна позволять выполнять следующие операции:

- регистрация пользователей, организаций,
- получение, редактирование и удаление информации о пользователях и организациях,
- регистрация операторов и разграничение их полномочий,
- создание, редактирование и удаление вакансий,
- отправка резюме пользователями,
- обработка резюме операторами,
- выбор вакансий, соответствующих заданных критериям,
- выбор резюме, соответствующих определённых критериям.

Задачи по разработке информационной системы:

- определение исходных отношений для заданной предметной области,
- нормализация полученных отношений,
- создание схемы базы данных,
- разработка программного продукта с реализацией CRUD.

Ход выполнения нормализации

С связи с большим количеством сущностей и связей было принято решение составить схему базы данных, изначально удовлетворяющую требованиям третьей нормальной формы. После анализа сущностей была проведена декомпозиция некоторых сущностей, сформирован список атрибутов каждой сущности и выполнена проверка соблюдения требований 3НФ.

Схема базы данных состоит из следующих сущностей:

- BRANCHES (ФИЛИАЛЫ),
- OPERATORS (ОПЕРАТОРЫ),
- RESUMES (РЕЗЮМЕ),
- VACANCIES (ВАКАНСИИ),
- COMPANIES (КОМПАНИИ),
- USERS (ПОЛЬЗОВАТЕЛИ),
- VACANCY_FIELDS (РАЗДЕЛЫ_ВАКАНСИЙ),
- DEGREES (УРОВНИ_ОБРАЗОВАНИЯ),
- FIELDS (ОБЛАСТИ_ОБРАЗОВАНИЯ),
- LANGUAGES (ЯЗЫКИ),
- KLADR_CACHE (КЭШ АДРЕСОВ KLADR ID).

Объяснение выбранной СУБД

Для реализации проекта была выбрана реляционная СУБД MySQL.

Реляционная база данных выбрана по следующим причинам:

- данные являются структурированными, их структура не подвергается частым изменениям;
- имеется большое количество связей между отношениями;
- реляционная СУБД обеспечивает независимость данных от интерфейса.

Описание сущностей

BRANCHES (ФИЛИАЛЫ) – включает информацию о филиалах Центра. Атрибут region_code содержит идентификатор адреса в системе КЛАДР. Характеризуется названием, кодом региона и адресом.

OPERATORS (ОПЕРАТОРЫ) – включается информацию об операторах. Операторы работают в филиалах (атрибут branch_id). Они имеют разные полномочия, для разграничения которых введён атрибут rights. Характеризуется филиалом, типом (главный оператор, оператор, администратор), фамилией, именем и отчеством, номером телефона, полномочиями. Атрибуты created_at, updated_at определяют время создания и редактирования записи.

RESUMES (РЕЗЮМЕ) – содержит информацию о резюме. Резюме принадлежит одному пользователю (user_id), размещается оператором (operator_id) определённого филиала (branch_id). Разделы вакансий описаны в другой таблице, резюме включает ID раздела. Также характеризуется вакансией, типом заработной платы (постоянная или сдельная), размером зарплаты, кодом региона в системе КЛАДР, опытом работы, занятостью (полная или неполная), графиком работы, его текстовым описанием, описанием

опыта работы. Атрибуты created_at, updated_at определяют время создания и редактирования записи, resume_lifetime – время хранения записи, active – активность резюме, contact – связь через оператора, public – доступность записи в списке.

VACANCIES (ВАКАНСИИ) – содержит информацию о вакансиях на бирже. Вакансия открывается одной организацией (company_id), размещается в системе, оператором (operator_id) и относится к определённому разделу (field_id). Также характеризуется названием, местом работы, параметром типа зарплаты (договорная, фиксированная), диапазоном размера зарплаты, типом зарплаты (постоянная или сдельная), наличием испытательного срока, графиком работы, его текстовым описанием, требованиями к соискателям, ближайшей к месту работы станцией метро, типом работы (постоянная или временная), описанием типа работы, должностными обязанностями, другими требованиями к соискателю (пол, возраст, образование, опыт работы, язык, уровень владения компьютером, специальность), дополнительной информацией. Атрибуты created_at, updated_at определяют время создания и редактирования записи, active – активность вакансии, date_edit – время актуальности вакансии, contact – связь через оператора, public – доступность записи в списке.

COMPANIES (КОМПАНИИ) – содержит информацию об организациях, которые размещают вакансии и просматривают резюме. Атрибут region_code содержит идентификатор адреса в системе КЛАДР. Характеризуется названием, кодом региона и адресом.

USERS (ПОЛЬЗОВАТЕЛИ) – содержит информацию о пользователях – соискателях. Область образования (edu1_field_id, edu2_field_id) соответствует значению из таблицы FIELDS. Уровень образования (edu1_degree_id, edu2_degree_id) соответствует значению из таблицы DEGREES. Язык (language_id) соответствует значению из таблицы LANGUAGES. Имеет следующие атрибуты: имя пользователя, пол, дата рождения, код города в системе КЛАДР, номер телефона, гражданство, наличие прописки, для двух образований – название учебного заведения, год окончания учебного заведения, специальность, область и уровень образования, форма обучения. Для хранения информации о дополнительном образовании добавлен атрибут edu_additional. Также характеризуется следующими атрибутами: язык, уровень владения языком, уровень владения компьютером, наличие водительского удостоверения, знания и навыки.

VACANCY_FIELDS (РАЗДЕЛЫ_ВАКАНСИЙ) – содержит информацию о разделах вакансий, перечисляет возможные значения.

DEGREES (УРОВНИ_ОБРАЗОВАНИЯ) – содержит информацию об уровнях образования, перечисляет возможные значения.

FIELDS (ОБЛАСТИ_ОБРАЗОВАНИЯ) – содержит информацию об областях образования, перечисляет возможные значения.

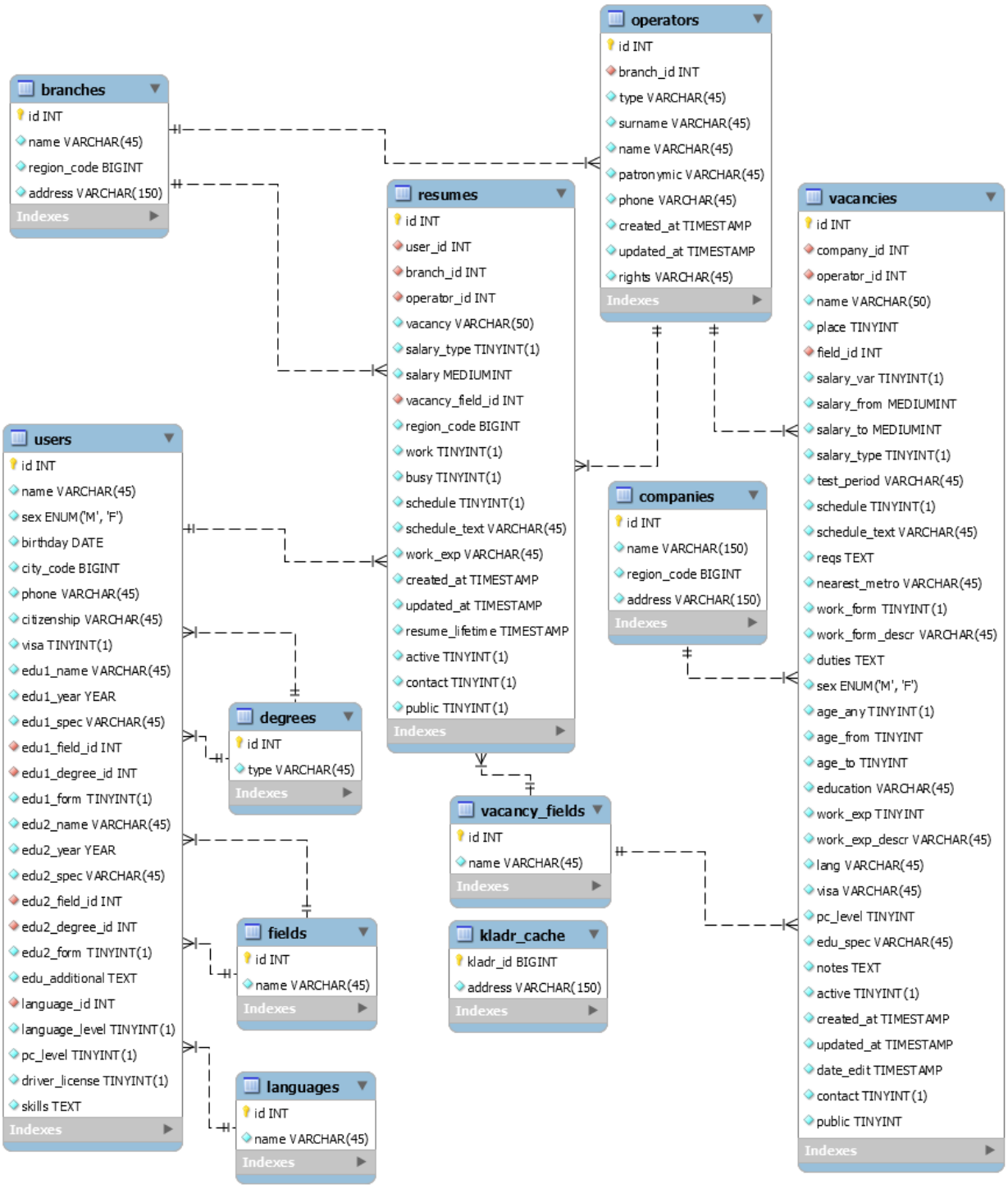
LANGUAGES (ЯЗЫКИ) – содержит информацию о языках, перечисляет возможные значения.

KLADR_CACHE (КЭШ АДРЕСОВ KLADR ID) – содержит информацию об адресах в системе КЛАДР, по которым ранее производились обращения к KLADR API, т.е. реализует кэш значений, соответствующих идентификаторам в системе.

Установлены следующие связи между сущностями:

Родительская сущность	**Атрибут**	**Дочерняя сущность**	**Внешний ключ**	**Тип связи**
BRANCHES	id	OPERATORS	branch_id	1:N
BRANCHES	id	RESUMES	branch_id	1:N
OPERATORS	id	RESUMES	operator_id	1:N
OPERATORS	id	VACANCIES	operator_id	1:N
COMPANIES	id	VACANCIES	company_id	1:N
USERS	id	RESUMES	user_id	1:N
VACANCY_FIELDS	id	RESUMES	vacancy_field_id	1:N
VACANCY_FIELDS	id	VACANCIES	field_id	1:N
DEGREES	id	USERS	edu1_degree_id	1:N
DEGREES	id	USERS	edu2_degree_id	1:N
FIELDS	id	USERS	edu1_field_id	1:N
FIELDS	id	USERS	edu2_field_id	1:N
LANGUAGES	id	USERS	language_id	1:N

ER-диаграмма



Реализация REST API

Для реализации REST API был настроен сервер со следующим программным обеспечением: Ubuntu Server 20.04, Apache, MySQL, PHP.

Серверная часть написана на языке программирования PHP и реализована с использованием Fat Free Framework 3.7.

Взаимодействие с базой данных осуществляется с помощью технологии ORM – Object-Relational Mapping. Используется встроенная в Fat Free Framework ORM – SQL Mapper.

Реализованы операции CRUD. Взаимодействие с серверной частью осуществляется путём отправки запросов по следующим маршрутам:

GET /@data – (READ, SQL-операция SELECT),
POST /@data/insert (CREATE, SQL-операция INSERT),
POST /@data/update/@id (UPDATE, SQL-операция UPDATE),
POST /@data/delete/@id (DELETE, SQL-операция DELETE).

@data – название таблицы, @id – id записи в таблице.

При запросе методом GET параметры querystring определяют критерии поиска. При отправке запросов методом POST добавляемые или редактируемые поля указываются в теле запроса.

Сервер возвращает результат выполнения операции в формате JSON. Пример успешного выполнения запроса выборки данных:


```
[
  {
    "id": 1,
    "name": "Sunshine Software",
    "region_code": 0,
    "address": "St. Peterburg"
  },
  {
    "id": 2,
    "name": "RapidSoft",
    "region_code": 1,
    "address": "Moscow"
  },
  {
    "id": 6,
    "name": "Native Development",
    "region_code": 0,
    "address": "Gatchina"
  }
]
```

Пример ответа сервера при возникновении ошибки:

```
{
    "status": "failed",
    "error_code": 1364,
    "error_string": "Field 'name' doesn't have a default value"
}
```

Некоторые таблицы имеют поля с ID, связанные с таблицами, которые содержат перечисления. Таблицы-перечисления: DEGREES, FIELDS, LANGUAGES, VACANCY_FIELDS. В данном случае использование типа данных ENUM нецелесообразно, т.к. количество вариантов может быть достаточно большим, при этом варианты могут изменяться.

При использовании связанных таким образом таблиц при выборке данных одной сущности для получения значений, указанных в виде ID в таблице перечислений, требуется выполнить дополнительные SQL-запросы. Чтобы избежать выполнения дополнительных запросов, для таблицы USERS используется представление USERS_DATA. Это позволяет получить полную информацию о пользователе отправкой только одного запроса.

Исходный текст запросов

Запрос по созданию таблиц

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO
_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema embit-dev
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema embit-dev
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `embit-dev` DEFAULT CHARACTER SET utf8 ;
USE `embit-dev` ;

-- -----------------------------------------------------
-- Table `embit-dev`.`branches`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `embit-dev`.`branches` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
```

```
  `region_code` BIGINT NOT NULL,
  `address` VARCHAR(150) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `embit-dev`.`operators`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `embit-dev`.`operators` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `branch_id` INT NOT NULL,
  `type` VARCHAR(45) NOT NULL,
  `surname` VARCHAR(45) NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  `patronymic` VARCHAR(45) NOT NULL,
  `phone` VARCHAR(45) NOT NULL,
  `created_at` TIMESTAMP NOT NULL,
  `updated_at` TIMESTAMP NOT NULL,
  `rights` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `branch_id_idx` (`branch_id` ASC) VISIBLE,
  CONSTRAINT `op_branch_id`
    FOREIGN KEY (`branch_id`)
    REFERENCES `embit-dev`.`branches` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `embit-dev`.`companies`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `embit-dev`.`companies` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(150) NOT NULL,
  `region_code` BIGINT NOT NULL,
  `address` VARCHAR(150) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `embit-dev`.`fields`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `embit-dev`.`fields` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `embit-dev`.`degrees`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `embit-dev`.`degrees` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `type` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `embit-dev`.`languages`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `embit-dev`.`languages` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `embit-dev`.`users`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `embit-dev`.`users` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  `sex` ENUM('M', 'F') NOT NULL,
  `birthday` DATE NOT NULL,
  `city_code` BIGINT NOT NULL,
  `phone` VARCHAR(45) NOT NULL,
  `citizenship` VARCHAR(45) NOT NULL,
  `visa` TINYINT(1) NOT NULL,
  `edu1_name` VARCHAR(45) NOT NULL,
  `edu1_year` YEAR NOT NULL,
  `edu1_spec` VARCHAR(45) NOT NULL,
  `edu1_field_id` INT NOT NULL,
  `edu1_degree_id` INT NOT NULL,
  `edu1_form` TINYINT(1) NOT NULL,
  `edu2_name` VARCHAR(45) NOT NULL,
  `edu2_year` YEAR NOT NULL,
  `edu2_spec` VARCHAR(45) NOT NULL,
  `edu2_field_id` INT NOT NULL,
  `edu2_degree_id` INT NOT NULL,
  `edu2_form` TINYINT(1) NOT NULL,
  `edu_additional` TEXT NOT NULL,
  `language_id` INT NOT NULL,
  `language_level` TINYINT(1) NOT NULL,
  `pc_level` TINYINT(1) NOT NULL,
  `driver_license` TINYINT(1) NOT NULL,
  `skills` TEXT NOT NULL,
  PRIMARY KEY (`id`),
```

```
  INDEX `edu1_field_id_idx` (`edu1_field_id` ASC) VISIBLE,
  INDEX `edu2_field_id_idx` (`edu2_field_id` ASC) VISIBLE,
  INDEX `edut1_degree_id_idx` (`edu1_degree_id` ASC) VISIBLE,
  INDEX `edu2_degree_id_idx` (`edu2_degree_id` ASC) VISIBLE,
  INDEX `language_id_idx` (`language_id` ASC) VISIBLE,
  CONSTRAINT `edu1_field_id`
    FOREIGN KEY (`edu1_field_id`)
    REFERENCES `embit-dev`.`fields` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `edu2_field_id`
    FOREIGN KEY (`edu2_field_id`)
    REFERENCES `embit-dev`.`fields` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `edu1_degree_id`
    FOREIGN KEY (`edu1_degree_id`)
    REFERENCES `embit-dev`.`degrees` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `edu2_degree_id`
    FOREIGN KEY (`edu2_degree_id`)
    REFERENCES `embit-dev`.`degrees` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `language_id`
    FOREIGN KEY (`language_id`)
    REFERENCES `embit-dev`.`languages` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `embit-dev`.`vacancy_fields`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `embit-dev`.`vacancy_fields` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `embit-dev`.`resumes`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `embit-dev`.`resumes` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `user_id` INT NOT NULL,
  `branch_id` INT NOT NULL,
  `operator_id` INT NOT NULL,
```

```
  `vacancy` VARCHAR(50) NOT NULL,
  `salary_type` TINYINT(1) NOT NULL,
  `salary` MEDIUMINT NOT NULL,
  `vacancy_field_id` INT NOT NULL,
  `region_code` BIGINT NOT NULL,
  `work` TINYINT(1) NOT NULL,
  `busy` TINYINT(1) NOT NULL,
  `schedule` TINYINT(1) NOT NULL,
  `schedule_text` VARCHAR(45) NOT NULL,
  `work_exp` VARCHAR(45) NOT NULL,
  `created_at` TIMESTAMP NOT NULL,
  `updated_at` TIMESTAMP NOT NULL,
  `resume_lifetime` TIMESTAMP NOT NULL,
  `active` TINYINT(1) NOT NULL,
  `contact` TINYINT(1) NOT NULL,
  `public` TINYINT(1) NOT NULL,
  PRIMARY KEY (`id`),
  INDEX `user_id_idx` (`user_id` ASC) VISIBLE,
  INDEX `branch_id_idx` (`branch_id` ASC) VISIBLE,
  INDEX `operator_id_idx` (`operator_id` ASC) VISIBLE,
  INDEX `vacancy_field_id_idx` (`vacancy_field_id` ASC) VISIBLE,
  CONSTRAINT `user_id`
    FOREIGN KEY (`user_id`)
    REFERENCES `embit-dev`.`users` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `branch_id`
    FOREIGN KEY (`branch_id`)
    REFERENCES `embit-dev`.`branches` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `operator_id`
    FOREIGN KEY (`operator_id`)
    REFERENCES `embit-dev`.`operators` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `vacancy_field_id`
    FOREIGN KEY (`vacancy_field_id`)
    REFERENCES `embit-dev`.`vacancy_fields` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `embit-dev`.`vacancies`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `embit-dev`.`vacancies` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `company_id` INT NOT NULL,
  `operator_id` INT NOT NULL,
```

```
`name` VARCHAR(50) NOT NULL,
`place` TINYINT NOT NULL,
`field_id` INT NOT NULL,
`salary_var` TINYINT(1) NOT NULL,
`salary_from` MEDIUMINT NOT NULL,
`salary_to` MEDIUMINT NOT NULL,
`salary_type` TINYINT(1) NOT NULL,
`test_period` VARCHAR(45) NOT NULL,
`schedule` TINYINT(1) NOT NULL,
`schedule_text` VARCHAR(45) NOT NULL,
`reqs` TEXT NOT NULL,
`nearest_metro` VARCHAR(45) NOT NULL,
`work_form` TINYINT(1) NOT NULL,
`work_form_descr` VARCHAR(45) NOT NULL,
`duties` TEXT NOT NULL,
`sex` ENUM('M', 'F') NOT NULL,
`age_any` TINYINT(1) NOT NULL,
`age_from` TINYINT NOT NULL,
`age_to` TINYINT NOT NULL,
`education` VARCHAR(45) NOT NULL,
`work_exp` TINYINT NOT NULL,
`work_exp_descr` VARCHAR(45) NOT NULL,
`lang` VARCHAR(45) NOT NULL,
`visa` VARCHAR(45) NOT NULL,
`pc_level` TINYINT NOT NULL,
`edu_spec` VARCHAR(45) NOT NULL,
`notes` TEXT NOT NULL,
`active` TINYINT(1) NOT NULL,
`created_at` TIMESTAMP NOT NULL,
`updated_at` TIMESTAMP NOT NULL,
`date_edit` TIMESTAMP NOT NULL,
`contact` TINYINT(1) NOT NULL,
`public` TINYINT NOT NULL,
PRIMARY KEY (`id`),
INDEX `company_id_idx` (`company_id` ASC) VISIBLE,
INDEX `operator_id_idx` (`operator_id` ASC) VISIBLE,
INDEX `field_id_idx` (`field_id` ASC) VISIBLE,
CONSTRAINT `company_id`
  FOREIGN KEY (`company_id`)
  REFERENCES `embit-dev`.`companies` (`id`)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
CONSTRAINT `vacancy_operator_id`
  FOREIGN KEY (`operator_id`)
  REFERENCES `embit-dev`.`operators` (`id`)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
CONSTRAINT `field_id`
  FOREIGN KEY (`field_id`)
  REFERENCES `embit-dev`.`vacancy_fields` (`id`)
  ON DELETE CASCADE
```

```
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `embit-dev`.`kladr_cache`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `embit-dev`.`kladr_cache` (
  `kladr_id` BIGINT NOT NULL,
  `address` VARCHAR(150) NOT NULL,
  PRIMARY KEY (`kladr_id`))
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Запрос по созданию представления

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `admin`@`%`
    SQL SECURITY DEFINER
VIEW `users_data` AS
    SELECT
        `users`.`id` AS `id`,
        `users`.`name` AS `name`,
        `users`.`sex` AS `sex`,
        `users`.`birthday` AS `birthday`,
        `users`.`city_code` AS `city_code`,
        `users`.`phone` AS `phone`,
        `users`.`citizenship` AS `citizenship`,
        `users`.`visa` AS `visa`,
        `users`.`edu1_name` AS `edu1_name`,
        `users`.`edu1_year` AS `edu1_year`,
        `users`.`edu1_spec` AS `edu1_spec`,
        `users`.`edu1_field_id` AS `edu1_field_id`,
        `f1`.`name` AS `edu1_field`,
        `users`.`edu1_degree_id` AS `edu1_degree_id`,
        `d1`.`type` AS `edu1_degree`,
        `users`.`edu1_form` AS `edu1_form`,
        `users`.`edu2_name` AS `edu2_name`,
        `users`.`edu2_year` AS `edu2_year`,
        `users`.`edu2_spec` AS `edu2_spec`,
        `users`.`edu2_field_id` AS `edu2_field_id`,
        `f2`.`name` AS `edu2_field`,
        `users`.`edu2_degree_id` AS `edu2_degree_id`,
        `d2`.`type` AS `edu2_degree`,
```

```
    `users`.`edu2_form` AS `edu2_form`,
    `users`.`edu_additional` AS `edu_additional`,
    `users`.`language_id` AS `language_id`,
    `languages`.`name` AS `language`,
    `users`.`language_level` AS `language_level`,
    `users`.`pc_level` AS `pc_level`,
    `users`.`driver_license` AS `driver_license`,
    `users`.`skills` AS `skills`
FROM
    (((((`users`
    JOIN `degrees` `d1` ON ((`users`.`edu1_degree_id` = `d1`.`id`)))
    JOIN `degrees` `d2` ON ((`users`.`edu2_degree_id` = `d2`.`id`)))
    JOIN `fields` `f1` ON ((`users`.`edu1_field_id` = `f1`.`id`)))
    JOIN `fields` `f2` ON ((`users`.`edu2_field_id` = `f2`.`id`)))
    JOIN `languages` ON ((`users`.`language_id` = `languages`.`id`)))
```

Заключение

В ходе работы была разработана база данных для Регионального центра содействия трудоустройству и адаптации к рынку труда выпускников.

Информационная система позволяет выполнять следующие операции: добавление, редактирование и удаление данных, получение записей, удовлетворяющих заданным условиям.

Был разработан программный проект с реализацией CRUD – сервис REST API.

Мельников Ф. В. _____