МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Информационные системы и технологии

Специальность 1-98 01 03 «Программное обеспечение информационной

безопасности мобильных систем»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

Разработка программного средства «Отдел кадров»

Выполнил студент Славин Никита Владимирович

(Ф.И.О.)

Руководитель проекта асс. Северинчик Н.А.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Заведующий кафедрой д.к.т.н., Смелов В.В.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Консультанты асс. Северинчик Н.А.

(учен. степень, звание, должность, подпись, Ф.И.О.)

(учен. степень, звание, должность, подпись, Ф.И.О.)

Нормоконтроллер асс. Северинчик Н.А.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Минск 2020

**Реферат**

Пояснительная записка курсового проекта содержит 21 страницы, 7 изображений, 4 источника литературы, 2 приложения.

Основными целями курсового проекта являются: разработка программного средства – «Отдел кадров».

Пояснительная записка состоит из введения, пяти разделов, заключения.

Во введении представлена общая информация, дающая представление о предстоящей работе, определены цели.

В первом разделе представлена постановка задачи.

Во втором разделе рассматривается разработка архитектуры проекта.

В третьем разделе рассматривается разработка функциональной модели приложения.

В четвертом разделе представлено руководство пользователя.

В заключении представлены итоги курсового проектирования и задачи, которые были решены в ходе разработки программного средства.

Содержание

[Введение 6](#_Toc515317847)

[1. Постановка задачи. 7](#_Toc515317848)

[2. Архитектура проекта 9](#_Toc515317849)

[3. Схема базы данных 10](#_Toc515317850)

[4. Программная реализация приложения 17](#_Toc515317851)

[4.1 Реализация сервера 17](#_Toc515317852)

[4.2 Реализация мобильного приложения 17](#_Toc515317853)

[4.2.1 REST-клиент 18](#_Toc515317854)

[5. Руководство пользователя 21](#_Toc515317855)

[5.1 Android приложение 21](#_Toc515317856)

[Заключение 27](#_Toc515317857)

[Список использованных источников 28](#_Toc515317858)

[Приложение А 29](#_Toc515317859)

[Приложение B 29](#_Toc515317860)

# Введение

Отдел кадров – это структура в организации, которая занимается управлением персоналом.

Отдел кадров является не только функциональной единицей, это еще и лицом компании, так как именно в отделе кадров любой соискатель начинает знакомиться с организацией.

Цель отдела кадров – это способствование достижению целей предприятия (организации) путем обеспечения предприятия необходимыми кадрами и эффективного использования потенциала работников.

Основная задача отдела кадров - правильно учитывать работу сотрудников, определять количество рабочих, выходных и больничных дней для расчета зарплаты, отпусков и подачи сведений в бухгалтерию организации.

Целью курсового проекта является проектирование и реализация программного средства «Отдел кадров».

# Постановка задачи.

ASP.NET (Active Server Pages для .NET) — платформа разработки [веб-приложений](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5), в состав которой входит: [веб-сервисы](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B8%D1%81), программная инфраструктура, модель программирования, от компании [Майкрософт](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D0%B9%D0%BA%D1%80%D0%BE%D1%81%D0%BE%D1%84%D1%82). ASP.NET входит в состав платформы [.NET Framework](https://ru.wikipedia.org/wiki/Microsoft_.NET) и является развитием более старой технологии [Microsoft ASP](https://ru.wikipedia.org/wiki/Active_Server_Pages).

ASP.NET внешне во многом сохраняет схожесть с более старой технологией ASP, что позволяет разработчикам относительно легко перейти на ASP.NET. В то же время внутреннее устройство ASP.NET существенно отличается от ASP, поскольку она основана на платформе [.NET](https://ru.wikipedia.org/wiki/Microsoft_.NET) и, следовательно, использует все новые возможности, предоставляемые этой платформой.

Microsoft SQL Server — [система управления реляционными базами данных (РСУБД)](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94), разработанная корпорацией [Microsoft](https://ru.wikipedia.org/wiki/Microsoft). Основной используемый язык запросов — [Transact-SQL](https://ru.wikipedia.org/wiki/Transact-SQL), создан совместно Microsoft и [Sybase](https://ru.wikipedia.org/wiki/Sybase). Transact-SQL является реализацией стандарта [ANSI](https://ru.wikipedia.org/wiki/%D0%90%D0%BC%D0%B5%D1%80%D0%B8%D0%BA%D0%B0%D0%BD%D1%81%D0%BA%D0%B8%D0%B9_%D0%BD%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%B8%D0%BD%D1%81%D1%82%D0%B8%D1%82%D1%83%D1%82_%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BE%D0%B2)/[ISO](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D0%B4%D1%83%D0%BD%D0%B0%D1%80%D0%BE%D0%B4%D0%BD%D0%B0%D1%8F_%D0%BE%D1%80%D0%B3%D0%B0%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D0%BF%D0%BE_%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8) по структурированному языку запросов ([SQL](https://ru.wikipedia.org/wiki/SQL)) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

SQL Server Management Studio (SSMS) — утилита из [Microsoft SQL Server](https://ru.wikipedia.org/wiki/Microsoft_SQL_Server) 2005 и более поздних версий для конфигурирования, управления и [администрирования](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%BD%D0%BE%D0%B5_%D0%B0%D0%B4%D0%BC%D0%B8%D0%BD%D0%B8%D1%81%D1%82%D1%80%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) всех компонентов Microsoft SQL Server. Утилита включает [скриптовый](https://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D1%80%D0%B8%D0%BF%D1%82) редактор и графическую программу, которая работает с объектами и настройками сервера.

Главным инструментом SQL Server Management Studio является Object Explorer, который позволяет пользователю просматривать, извлекать объекты сервера, а также полностью ими управлять.

ADO.NET Entity Framework (EF) — объектно-ориентированная технология доступа к данным, является [object-relational mapping](https://ru.wikipedia.org/wiki/ORM) (ORM) решением для [.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework) от [Microsoft](https://ru.wikipedia.org/wiki/Microsoft). Предоставляет возможность взаимодействия с объектами как посредством [LINQ](https://ru.wikipedia.org/wiki/LINQ) в виде LINQ to Entities, так и с использованием Entity SQL. Для облегчения построения web-решений используется как [ADO.NET Data Services](https://ru.wikipedia.org/wiki/ADO.NET_Data_Services) ([Astoria](https://ru.wikipedia.org/wiki/ADO.NET_Data_Services)), так и связка из [Windows Communication Foundation](https://ru.wikipedia.org/wiki/Windows_Communication_Foundation) и [Windows Presentation Foundation](https://ru.wikipedia.org/wiki/Windows_Presentation_Foundation), позволяющая строить многоуровневые приложения, реализуя один из шаблонов проектирования [MVC](https://ru.wikipedia.org/wiki/Model-View-Controller), [MVP](https://ru.wikipedia.org/wiki/Model-View-Presenter) или [MVVM](https://ru.wikipedia.org/wiki/Model-View-ViewModel).

При разработке сервера следует реализовать следующие API:

* REST – это стиль архитектуры программного обеспечения для распределенный систем, таких как WorldWideWeb, который, как правило используется для построения веб-служб.
* SOAP – протокол обмена структурированными сообщениями в распределенной вычислительной среде. Первоначально предназначался для реализации удаленного вызова процедур, сейчас используется для обмена сообщениями в формате XML или JSON.

# Архитектура проекта

Распределенное приложение состоит из 2-х частей:

Сервер. Данная часть приложения предоставляет REST API и модуль SOAP. Доступ к данным базы данных осуществляется с помощью CRUD операций.

Мобильное приложение. Эта часть использует RESTClient для синхронизации информации локальной базы данных мобильного приложения с удаленной базой данных. SOAPRequestservice для проверки соединения мобильного клиента с сервером.

Графическое представление архитектуры распределенного приложения изображено на рисунке 2.1.

**Android Studio:**

* список сотрудников
* меню выбора активности

**Java class:**

* Клиент
* Выгрузить изб. работы запись

**ASP.NET:**

* HTTP-API
* Выдача записей

**EF Model:**

* Пользователи

**Java class:**

* Авторизация
* Регистрация

**Java class:**

* Вывод сотрудников
* Вывод изб. сотрудников

**MS SQL:**

* База данных

**C# class:**

* Администрирование пользователей
* Добавление новых пользователей

Рисунок. 2.1 – архитектура распределенного приложения

# Схема базы данных

В ходе написания приложения была разработана база данных «HumanResource»схема которой представлена на рисунке 3.1.

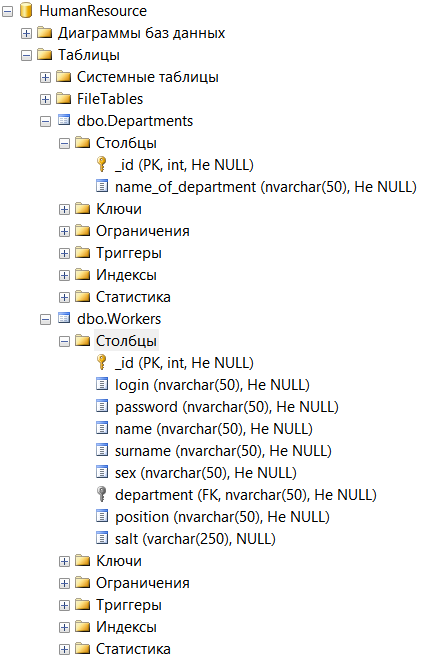


Рисунок 3.1 – структура базы данных

Таблица Departments хранит в себе данные об отделах, присутствующих в организации. Таблица Workers хранит в себе всю информацию о работниках (Их логины, пароли, имена, фамилии и т.д.)

# Программная реализация приложения

## 4.1 Реализация сервера

Поскольку ASP.NET основывается на [Common Language Runtime](https://ru.wikipedia.org/wiki/Common_Language_Runtime) (CLR), которая является основой всех приложений Microsoft .NET, разработчики могут писать код для ASP.NET, используя языки программирования, входящие в комплект [.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework) ([C#](https://ru.wikipedia.org/wiki/C_Sharp), [Visual Basic.NET](https://ru.wikipedia.org/wiki/VB.NET), J# и [JScript .NET](https://ru.wikipedia.org/wiki/JScript_.NET)). ASP.NET имеет преимущество в скорости по сравнению со скриптовыми технологиями, так как при первом обращении код компилируется и помещается в специальный [кэш](https://ru.wikipedia.org/wiki/%D0%9A%D1%8D%D1%88), и впоследствии только исполняется, не требуя затрат времени на синтаксический анализ, [оптимизацию](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D1%82%D0%B8%D0%BC%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D0%BA%D0%BE%D0%BC%D0%BF%D0%B8%D0%BB%D1%8F%D1%82%D0%BE%D1%80%D0%B0) и т. д.

Программная модель ASP.NET основывается на протоколе [HTTP](https://ru.wikipedia.org/wiki/HTTP) и использует его правила взаимодействия между сервером и браузером. При формировании страницы заложена абстрактная программной модель Web Forms, и на ней основана основная часть реализации программного кода.

Сервер имеет следующие пути для обработки HTTP-запросов (Приложение А)

* Get – получение пользователей;
* Delete – для удаления пользователя;
* Post – добавление пользователей;

## 4.2 Реализация мобильного приложения

Мобильное приложение реализовано для операционной системы Android 6.0 и выше. Данное приложение было разработано в специальной среде разработки –Android Studio.

Графическая структура классов представлена на рисунке 4.2.1.

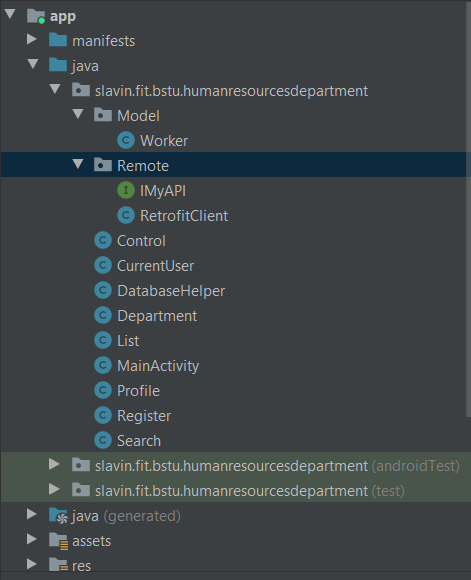


Рисунок 4.2.1 – структура проекта

### Проект в себе содержит папку Model, которая необходима для создания объектов Worker для помещения их в базу данных, в ней реализуется класс:

1. Worker – java класс с помощью которого мы будем создавать объекты пользователей и заносить их в локальную и удалённую бд;

Remote содержит в себе интерфейс и класс для работы с сервером:

1. IMyAPI – Интерфейс с набором методов для работы с сервером;
2. RetrofitClient – класс в котором мы реализуем методы для работы с сервером;

Дальше у нас идут просто Java-классы для каждой активити присутствующей в приложении:

1. Control – панель управления;
2. CurrentUser – открыть профиль определённого пользователя;
3. Department – главная страница приложения;
4. List – список всех сотрудников;
5. MainActivity – страница логина;
6. Profile – профиль залогиненного сотрудница;
7. Register – форма регистрации;
8. Search – поиск сотрудников;

Также есть класс для работы с локальной базой данных:

1. DatabaseHelper – класс для работы с SQLite;

### **4.2.1 REST-клиент**

Для синхронизации мобильного приложения с сервером применяется архитектурный стиль REST. Основная задача сервера в этой архитектуре предоставить клиентам доступ к ресурсам по их идентификаторам. Под доступом подразумевается как получение информации, так и ее изменение или удаление. В зависимости от реализации сервиса, в ответ на наш запрос придёт ответ в формате JSON. На рисунке 4.2.1.1 представлен пример одного из get-запросов.

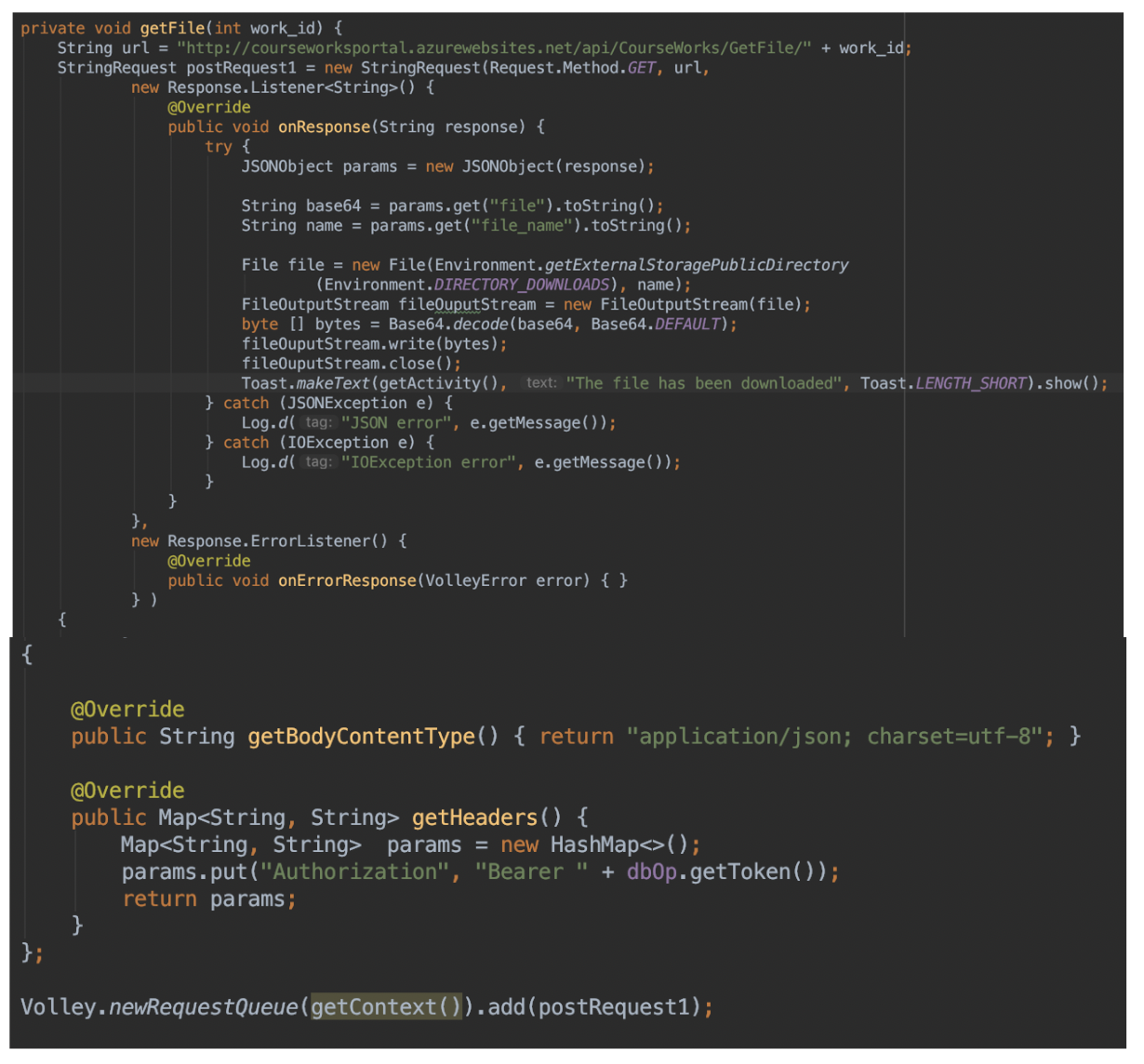


Рисунок 4.2.1.1 – get-запрос

Request метод указывает на то, какие операции будут производиться с ресурсом. GET – позволяет получить данные, а также конкретный ответ по заданным параметрам. POST – позволяет передать данные на сервер. Например, передача выбранного пользователя, для последующей записи в удаленную базу данных.

Таким образом можно сказать, что REST – это взаимодействие «клиент-серверной» архитектуры с помощью HTTP-запросов и HTTP-ответов по протоколу, не подтверждающему состояние.

# Руководство пользователя

## 5.1 Android приложение

Для начала необходимо установить приложение на мобильный устройство с API не меньше 23 (Android 6.0). Запустим наше приложение. Сразу после запуска мы увидим окно авторизации и регистрации (Рисунок 5.1.1). Авторизация происходит с помощью механизма JWT токена (JSON Web Token (JWT) — это открытый стандарт ([RFC 7519](https://tools.ietf.org/html/rfc7519)) для создания токенов доступа, основанный на формате [JSON](https://ru.wikipedia.org/wiki/JSON). Как правило, используется для передачи данных и [аутентификации](https://ru.wikipedia.org/wiki/%D0%90%D1%83%D1%82%D0%B5%D0%BD%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F) в клиент-серверных приложениях. JWT создаются сервером, подписываются секретным ключом и передаются клиенту, который в дальнейшем использует данный токен для подтверждения своей личности).

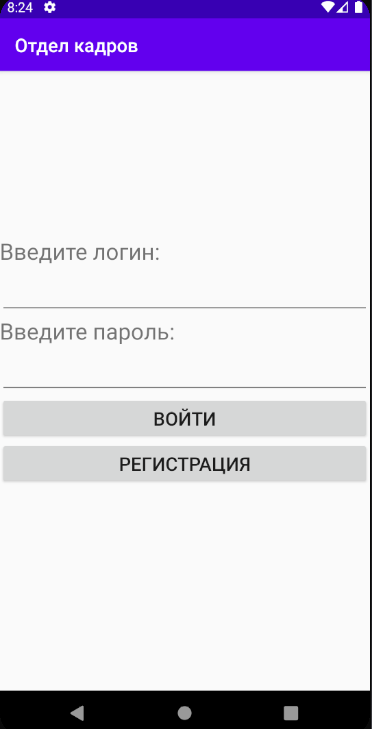


Рисунок 5.1.1 – окно авторизации и регистрации

После ввода данных (имя пользователя и пароль) мы можем видеть весь функционал приложения (Рисунок 5.1.2).

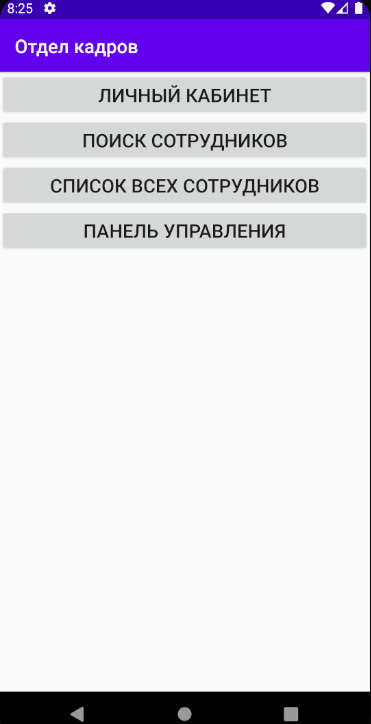
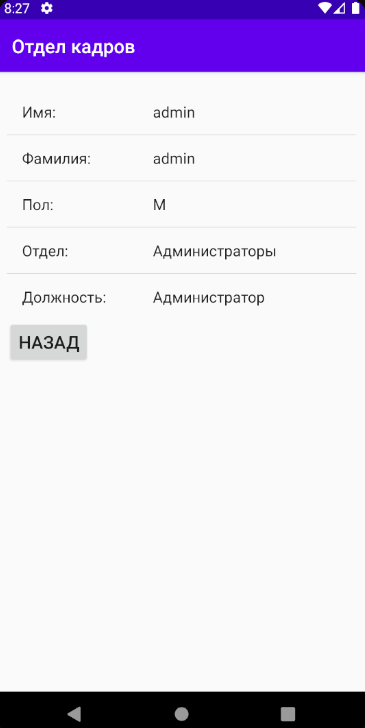


Рисунок 5.1.2 – главная страница

Для перехода на страницу профиля залогиненного сотрудника, используется кнопка «Личный кабинет» по нажатию на которую открывается новая активити. (Рисунок 5.1.3).



Риснок 5.1.3 – профиль

Для поиска нужного сотрудника по каким-то определённым критериям нужно нажать кнопку «Поиск сотрудников». Откроется новая активити в котором мы выбираем нужные нам критерии.

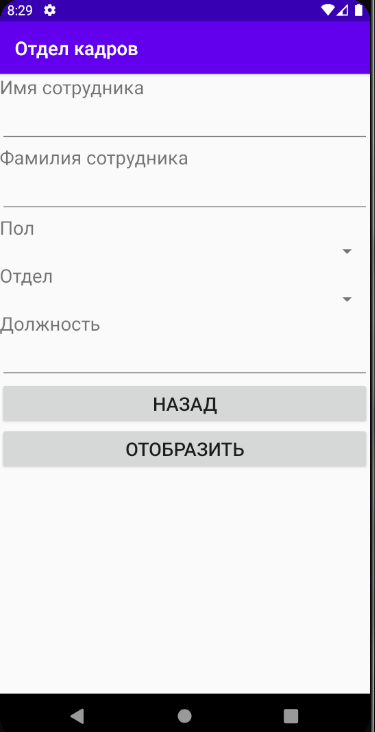


Рисунок 5.1.4 – поиск

Для просмотра всех сотрудников, работающих в организации, нажимаем на кнопку «Список всех сотрудников».

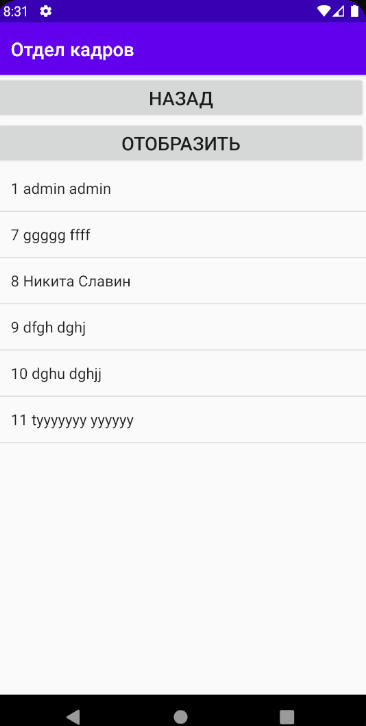


Рисунок 5.1.5 – список всех сотрудников

# Заключение

Решая поставленную задачу, я пришел к удовлетворительному результату. Таким образом, была достигнута цель, и был создан проект «Отдел кадров», который включает в себя android–клиент и сервер.

Программная система имеет простой, интуитивно понятный пользователю интерфейс.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанный продукт работает корректно, а требования технического задания выполнены в полном объеме.

В результате приложение обладает следующими особенностями:

* реализован качественный алгоритм подгрузки тяжелой информации;
* реализована отправка данных на удалённый сервер по технологии REST API;
* реализован полностью работоспособный сервер и удалённая база данных.
* реализована защита приложения в виде запрета на скриншоты и съёмку видео, а также авторизации.

# Список использованных источников

1. ASP.NET - https://docs.microsoft.com/ru-ru/aspnet;

2. Entity Framework - https://docs.microsoft.com/en-us/ef;

3. SQL Server Management Studio - https://docs.microsoft.com/en-us/sql/ssms;

4. Android documentation - https://developer.android.com/docs.

# Приложение А

using System;

using System.Collections.Generic;

using System.IdentityModel.Tokens.Jwt;

using System.Linq;

using System.Security.Claims;

using System.Threading.Tasks;

using CourseWorksPortal.Models;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using Microsoft.IdentityModel.Tokens;

using Newtonsoft.Json;

namespace CourseWorksPortal.Controllers

{

public class AccountJwtController : Controller

{

private DatabaseContext db;

public AccountJwtController(DatabaseContext context)

{

db = context;

}

[HttpPost("/token")]

public async Task Token()

{

var username = Request.Form["username"];

var password = Request.Form["password"];

User user = await db.Users.FirstOrDefaultAsync(u => u.Username == username && u.Password == password);

var identity = GetIdentity(user);

if (identity == null)

{

Response.StatusCode = 400;

await Response.WriteAsync("Invalid username or password");

return;

}

// создаем JWT-токен

var now = DateTime.UtcNow;

var jwt = new JwtSecurityToken(

issuer: AuthOptions.ISSUER,

audience: AuthOptions.AUDIENCE,

notBefore: now,

claims: identity.Claims,

expires: now.Add(TimeSpan.FromDays(AuthOptions.LIFETIME)),

signingCredentials: new SigningCredentials(AuthOptions.GetSymmetricSecurityKey(), SecurityAlgorithms.HmacSha256));

var encodedJwt = new JwtSecurityTokenHandler().WriteToken(jwt);

var response = new

{

access\_token = encodedJwt,

id = user.Id

};

// сериализация ответа

Response.ContentType = "application/json";

await Response.WriteAsync(JsonConvert.SerializeObject(response, new JsonSerializerSettings { Formatting = Formatting.Indented }));

}

private ClaimsIdentity GetIdentity(User user)

{

if (user != null)

{

var claims = new List<Claim>

{

new Claim(ClaimsIdentity.DefaultNameClaimType, user.Username),

new Claim(ClaimsIdentity.DefaultRoleClaimType, user.Role)

};

ClaimsIdentity claimsIdentity =

new ClaimsIdentity(claims, "Token", ClaimsIdentity.DefaultNameClaimType,

ClaimsIdentity.DefaultRoleClaimType);

return claimsIdentity;

}

// если пользователя не найдено

return null;

}

}

}

# Приложение B

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using CourseWorksPortal.Models;

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace CourseWorksPortal.Controllers.WebAPI

{

[Route("api/[controller]/[action]")]

[Authorize(AuthenticationSchemes = JwtBearerDefaults.AuthenticationScheme)]

public class CourseWorksController : Controller

{

DatabaseContext db;

public CourseWorksController(DatabaseContext context)

{

db = context;

}

[HttpGet]

public IEnumerable<dynamic> Get()

=> db.CourseWorks.Select(p => new { p.Id, p.Name, p.Description, p.File\_name }).ToList();

[HttpGet("{id}")]

public IActionResult Get(int id)

{

CourseWork courseWork = db.CourseWorks.FirstOrDefault(x => x.Id == id);

if (courseWork != null)

return new ObjectResult(courseWork);

return NotFound();

}

[HttpGet("{id}")]

public dynamic GetFile(int id)

{

return db.CourseWorks.Where(p => p.Id == id).Select(p => new { p.File, p.File\_name }).FirstOrDefault();

}

}

}