

МИНИСТЕРСТВО ОБРАЗОВАНИЯ, НАУКИ И МОЛОДЕЖНОЙ ПОЛИТИКИ  
НИЖЕГОРОДСКОЙ ОБЛАСТИ  
ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ  
«НИЖЕГОРОДСКИЙ РАДИОТЕХНИЧЕСКИЙ КОЛЛЕДЖ»

**ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

ПМ.02 Осуществление интеграции программных модулей  
(наименование модуля)

Обучающийся Абашин Антон Вячеславович  
(Ф.И.О.)

Специальность: 09.02.07 Информационные системы и программирование

Квалификация: Программист

Курс: 4

Группа: 4ИСиП-18-2

Руководитель практики от ГБПОУ «НРТК» преподаватель  
Должность

Щербатюк Марина Сергеевна  
(Ф.И.О.)

Нижний Новгород  
2022г.

## Содержание

### Оглавление

ВВЕДЕНИЕ .....	3
Выполнение индивидуального задания.....	4
1 Краткая история создания предприятия .....	4
2 Разработка мобильного приложения .....	5
2.1 Авторизация.....	6
2.2 Регистрация.....	8
3 Подключение базы данных .....	13

## **ВВЕДЕНИЕ**

Производственная практика была направлена на формирование компетенций:

- Выбирать способы решения задач профессиональной деятельности;
- Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности;
- Планировать и реализовывать собственное профессиональное и личностное развитие;
- Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами;
- Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста
- Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях.
- Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности
- Использовать информационные технологии в профессиональной деятельности.
- Пользоваться профессиональной документацией на государственном и иностранном языке.
- Планировать предпринимательскую деятельность в профессиональной сфере.
- Осуществлять инсталляцию, настройку и обслуживание программного обеспечения компьютерных систем.
- Осуществлять измерения эксплуатационных характеристик программного обеспечения компьютерных систем.
- Выполнять работы по модификации отдельных компонент программного обеспечения в соответствии с потребностями заказчика.
- Обеспечивать защиту программного обеспечения компьютерных систем программными средствами.

## Выполнение индивидуального задания

### 1 Краткая история создания предприятия

#### ОТКРЫТА ШКОЛА



Есть на нашей приволжской земле еще не одна сотня самых бесценных памятников выдающемуся народному; педагогу и просветителю И.Н. Ульянову – это школы, биография которых началась в «ульяновское время».

Болтинка бывшей Бахаревской волости Курмышского уезда, ныне Сеченовского района. 15 ноября 1872 год. Здесь, в новом здании, построенном крестьянами на свои средства, было открыто мужское начальное училище.

Илья Николаевич инспектировал его 23 ноября 1873 года. В день его приезда обучалось 70 мальчиков. Попечителем школы был крестьянин Н. Григорьев, учителем – выпускник духовного училища – Андрей Пальмов.

*Газета "Горьковская Правда" 26.07.1981г.*

#### ИЗ ИСТОРИИ НАШЕЙ ШКОЛЫ



В 1975 году Болтинский педагогический коллектив стал работать в новой школе. С вселением в новое здание началась напряженная работа педагогов и учащихся.

За время своего пребывания в новом здании были созданы необходимые условия для обучения. Болтинскую школу окончили сотни юношей и девушек. Многие из них закончили высшие и средние заведения, получив различные специальности. За все это время трудно назвать год, в который бы выпускники не поступали в высшие учебные заведения. И способствовали им в этом наши педагоги.

Выпускники Болтинской средней школы учатся и работают в разных местах и имеют различные специальности от технички, до руководителей районных и областных организаций и предприятий. И где бы они ни были, со своими обязанностями справляются успешно. В этом заслуга педагогического коллектива школы.

## 2 Разработка мобильного приложения

Согласно техническому заданию было разработано мобильное приложение, удовлетворяющее всем поставленным требованиям.

Структура любого приложения состоит из xml файлов и java классов.

Xml файл представляет собой layout resource file, на котором располагается вся визуальная составляющая мобильного приложения.

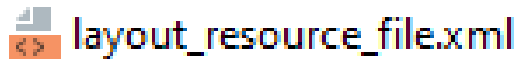


Рисунок 2.1. Layout Resource File

Функциональная составляющая приложения располагается в Java class.



Рисунок 2.2. Java Class

## 2.1 Авторизация.

Для получения доступа ко всем функциям мобильного приложения пользователь должен авторизоваться, заполнив все необходимые для этого поля. При успешной авторизации пользователь попадает в главное меню приложения.

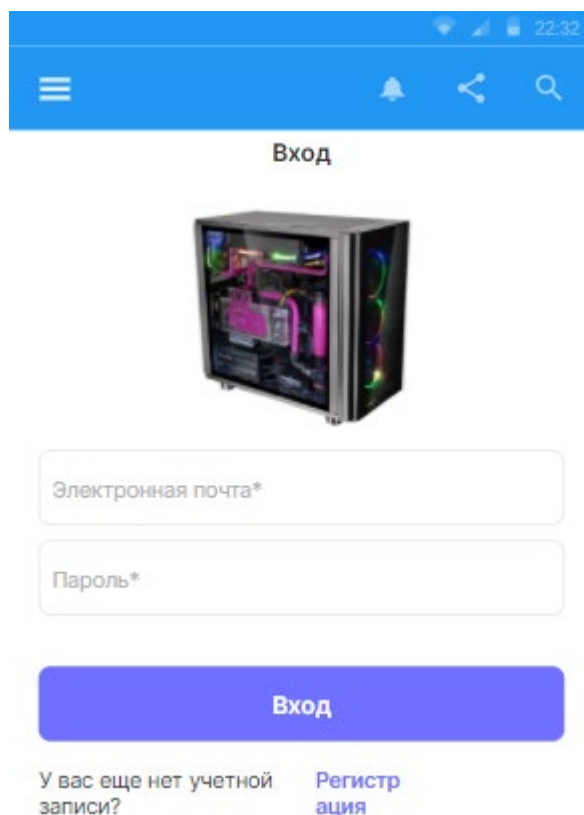


Рисунок 2.3 - Графический интерфейс "Авторизация"

На рисунке 2.4 представлен код для создания и привязки текстовых полей окна «Авторизация».

```
private void showSignInWindow() {  
    AlertDialog.Builder dialog = new AlertDialog.Builder(context: this);  
    dialog.setTitle("Войти");  
    dialog.setMessage("Введите данные для успешной авторизации");  
  
    LayoutInflater inflater = LayoutInflater.from(this);  
    View signin_window = inflater.inflate(R.layout.signin_window, root: null);  
  
    dialog.setView(signin_window);  
  
    final MaterialEditText Email = signin_window.findViewById(R.id.emailField);  
    final MaterialEditText password = signin_window.findViewById(R.id.passwordField);  
}
```

Рисунок 2.4 – Код окна "Авторизация".

```

dialog.setNegativeButton( text: "Отменить", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int which) {
        dialogInterface.dismiss();
    }
});

```

Рисунок 2.5 – Код кнопки "Отмена".

На рисунке 2.6 представлен код для кнопки «Войти», который сначала проверяет все поля окна «Авторизация» на правильность и полноту заполнения, затем производит авторизацию пользователя в приложение. При неверно введенных данных выдает ошибку и закрывает окно «Авторизация», возвращая на начальный экран приложения.

```

dialog.setPositiveButton( text: "Войти", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int which) {
        if (TextUtils.isEmpty(Email.getText().toString())) {
            Snackbar.make(root, text: "Вы не ввели Email", Snackbar.LENGTH_SHORT).show();
            return;
        }
        if (password.getText().toString().length() < 6) {
            Snackbar.make(root, text: "Вы ввели неверный пароль", Snackbar.LENGTH_SHORT).show();
            return;
        }
        auth.signInWithEmailAndPassword(Email.getText().toString(), password.getText().toString())
            .addOnSuccessListener(new OnSuccessListener<AuthResult>() {
                @Override
                public void onSuccess(AuthResult authResult) {
                    startActivity(new Intent( packageContext: ActivityAuthorization.this, MainActivity.class));
                    finish();
                }
            }).addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    Snackbar.make(root, text: "Ошибка авторизации", Snackbar.LENGTH_SHORT).show();
                }
            });
    }
});
dialog.show();

```

Рисунок 2.6 – Код кнопки "Войти".

## 2.2 Регистрация.

Для получения доступа к мобильному приложению пользователь должен зарегистрировать аккаунт, заполнив все необходимые для этого поля. При успешной регистрации данные отправляются в соответствующую таблицу базы данных.

Регистрация

Электронная почта\*

Пароль\*

Фамилия

Имя

Отчество

Дата рождения

Телефон

Регистрация

У вас уже есть учетная запись? [Вход](#)

Рисунок 2.7 - Графический интерфейс "Регистрация"



На рисунке 2.8 представлен код для создания и привязки текстовых полей окна «Регистрация».

```
private void showRegisterWindow() {
    AlertDialog.Builder dialog = new AlertDialog.Builder(context: this);
    dialog.setTitle("Регистрация");
    dialog.setMessage("Введите данные для регистрации:");

    LayoutInflater inflater = LayoutInflater.from(this);
    View register_window = inflater.inflate(R.layout.register_window, root: null);

    dialog.setView(register_window);

    final MaterialEditText Email = register_window.findViewById(R.id.emailField);
    final MaterialEditText phone = register_window.findViewById(R.id.phoneField);
    final MaterialEditText password = register_window.findViewById(R.id.passwordField);
    final MaterialEditText firstName = register_window.findViewById(R.id.firstNameField);
    final MaterialEditText lastName = register_window.findViewById(R.id.lastNameField);
    final MaterialEditText patronymic = register_window.findViewById(R.id.patronymicField);
    final MaterialEditText birthday = register_window.findViewById(R.id.birthdayField);
}
```

Рисунок 2.8 – Код окна "Регистрация".

На рисунке 2.9 представлен код для кнопки «Зарегистрироваться», который проверяет все поля окна «Регистрация» на правильность и полноту заполнения.

```

dialog.setPositiveButton( text: "Зарегистрироваться", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int which) {
        if (TextUtils.isEmpty(Email.getText().toString())) {
            Snackbar.make(root, text: "Вы не ввели Email", Snackbar.LENGTH_SHORT).show();
            return;
        }
        if (password.getText().toString().length() < 6) {
            Snackbar.make(root, text: "Пароль должен содержать не менее 6 символов!", Snackbar.LENGTH_SHORT).show();
            return;
        }
        if (TextUtils.isEmpty(firstName.getText().toString())) {
            Snackbar.make(root, text: "Вы не ввели имя", Snackbar.LENGTH_SHORT).show();
            return;
        }
        if (TextUtils.isEmpty(lastName.getText().toString())) {
            Snackbar.make(root, text: "Вы не ввели фамилию", Snackbar.LENGTH_SHORT).show();
            return;
        }
        if (TextUtils.isEmpty(patronymic.getText().toString())) {
            Snackbar.make(root, text: "Вы не ввели отчество", Snackbar.LENGTH_SHORT).show();
            return;
        }
        if (TextUtils.isEmpty(birthday.getText().toString())) {
            Snackbar.make(root, text: "Вы не ввели дату рождения", Snackbar.LENGTH_SHORT).show();
            return;
        }
        if (TextUtils.isEmpty(phone.getText().toString())) {
            Snackbar.make(root, text: "Вы не ввели телефон", Snackbar.LENGTH_SHORT).show();
            return;
        }
    }
});

```

Рисунок 2.9 - Код на проверку полей окна «Регистрация» для кнопки «Добавить».

```

dialog.setNegativeButton( text: "Отменить", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int which) {
        dialogInterface.dismiss();
    }
});

```

Рисунок 2.10 - Код для кнопки «Отменить».

Самая главная часть кода окна «Регистрация» представлена на рисунке 2.11, при помощи которой происходит непосредственно регистрация аккаунта и заполнение данных в таблицу «Users».

```

        auth.createUserWithEmailAndPassword(Email.getText().toString(), password.getText().toString())
            .addOnSuccessListener(new OnSuccessListener<AuthResult>() {
                @Override
                public void onSuccess(AuthResult authResult) {
                    Users user = new Users();
                    user.setEmail(Email.getText().toString());
                    user.setPassword(password.getText().toString());
                    user.setFirstName(firstName.getText().toString());
                    user.setLastName(lastName.getText().toString());
                    user.setPatronymic(patronymic.getText().toString());
                    user.setBirthday(birthday.getText().toString());
                    user.setPhone(phone.getText().toString());

                    users.child(FirebaseAuth.getInstance().getCurrentUser().getUid())
                        .setValue(user)
                        .addOnSuccessListener(new OnSuccessListener<Void>() {
                            @Override
                            public void onSuccess(Void aVoid) {
                                Snackbar.make(root, text: "Регистрация прошла успешно", Snackbar.LENGTH_SHORT).show();
                            }
                        });
                }
            });
    }
}

dialog.show();
}

```

Рисунок 2.11 - Код для регистрации аккаунта по нажатию кнопки «Зарегистрироваться».

На рисунке 2.12 представлен код Java Class «Users», который необходим для сохранения данных текстовых полей окна «Регистрация» и «Авторизация».

```
public class Users {  
    private String Email, password, firstName, lastName, patronymic, birthday, phone;  
    public Users() {}  
  
    public Users(String Email, String phone, String password, String firstName, String lastName, String patronymic, String birthday) {  
        this.Email = Email;  
        this.password = password;  
        this.firstName = firstName;  
        this.lastName = lastName;  
        this.patronymic = patronymic;  
        this.birthday = birthday;  
        this.phone = phone;  
    }  
    public String getEmail() { return Email; }  
  
    public void setEmail(String email) { this.Email = email; }  
  
    public String getPassword() { return password; }  
  
    public void setPassword(String password) { this.password = password; }  
  
    public String getFirstName() { return firstName; }  
  
    public void setFirstName(String firstName) { this.firstName = firstName; }  
  
    public String getLastName() { return lastName; }  
  
    public void setLastName(String lastName) { this.lastName = lastName; }  
  
    public String getPatronymic() { return patronymic; }  
  
    public void setPatronymic(String patronymic) { this.patronymic = patronymic; }  
  
    public String getBirthday() { return birthday; }  
  
    public void setBirthday(String birthday) { this.birthday = birthday; }  
  
    public String getPhone() { return phone; }  
  
    public void setPhone(String phone) { this.phone = phone; }  
}
```

Рисунок 2.12 - Код Java Class «Users»

### 3 Подключение базы данных

Для разработки баз данных была использована облачная база данных «Firebase», которая позволяет пользователям хранить и получать сохраненную информацию, а так же имеет удобные средства и методы взаимодействия с ней.

Firebase – это облачная база данных, которая позволяет пользователям хранить и получать сохраненную информацию, а также имеет удобные средства и методы взаимодействия с ней.

Firebase хранит текстовые данные в JSON формате и предоставляет удобные методы для чтения, обновления и извлечения данных. Также, Firebase может помочь с регистрацией и авторизацией пользователей, хранением сессий (авторизованные пользователи), медиафайлов к которым с легкостью предоставляет доступ благодаря Cloud Storage.


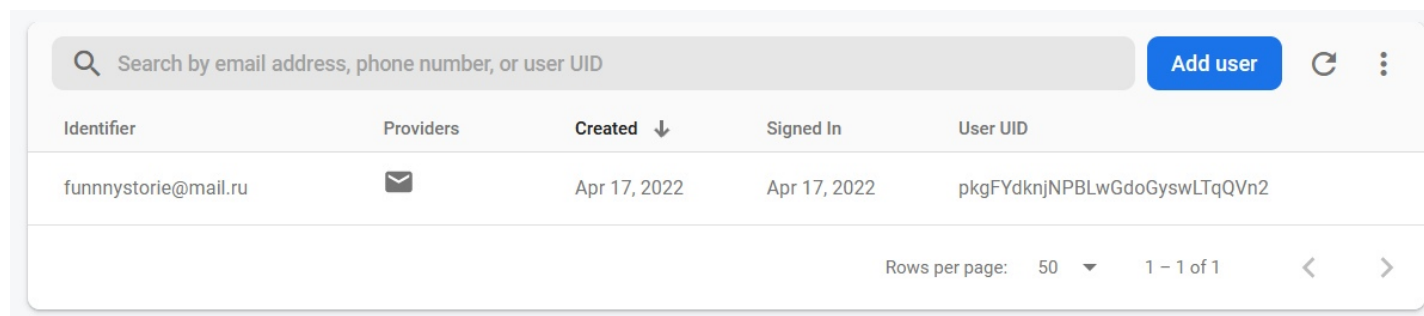

`https://valore-5705a-default-rtdb.firebaseio.com/`  
 Users

Рисунок 3.1 - Структура таблиц с данными в Firebase.



Identifier	Providers	Created ↓	Signed In	User UID
funnnystorie@mail.ru		Apr 17, 2022	Apr 17, 2022	pkgFYdknjNPBLwGdoGyswLTqQVn2

Rows per page: 50 1 – 1 of 1

Рисунок 3.2 - Окно «Authentication» Firebase.

На рисунке 4.3 представлена структура таблицы «Users», которая хранит в себе данные о пользователях, зарегистрировавших аккаунт в мобильном приложении.

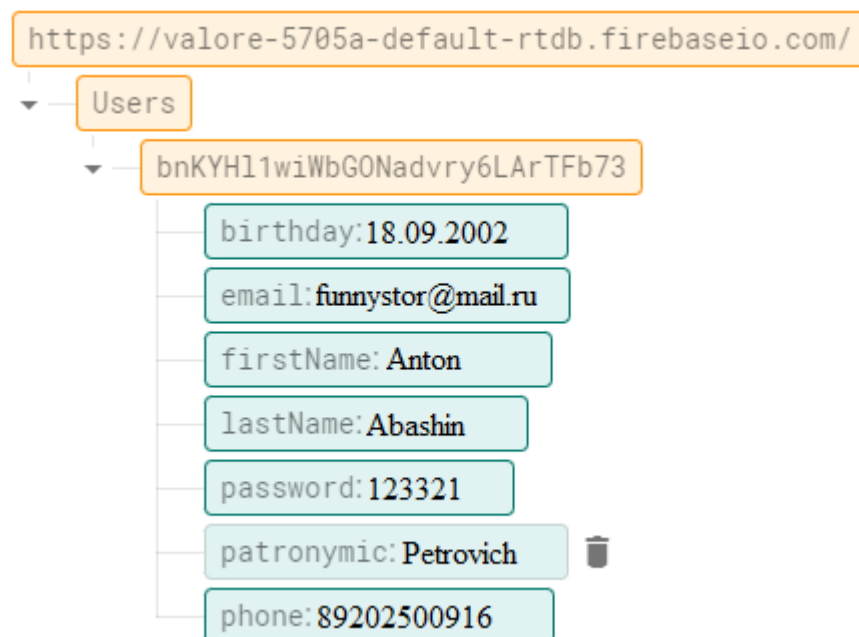


Рисунок 3.3 - Таблица «Users» с информацией о пользователях.

Обучающийся \_\_\_\_\_ / \_\_\_\_\_ /  
подпись расшифровка подписи