УДК 004.932.2

Еремин Данила Александрович,
студент 3 курса, факультет Компьютерных наук и технологий,
Восточно-Сибирский государственный университет технологий и управления
Жилин Данил Евгеньевич,
студент 3 курса, факультет Компьютерных наук и технологий,
Восточно-Сибирский государственный университет технологий и управления
Россия, г. Улан-Удэ

ПРИМЕНЕНИЕ QR-КОДОВ И ИДЕНТИФИКАЦИЯ ЧЕЛОВЕКА ПО ЛИЦУ НА КОНТРОЛЬНЫХ ПУНТКАХ

Аннотация: *в статье рассмотрена готовая система по идентификации человека с использованием языков программирования: Python, Kotlin, SQL.*

Ключевые слова: *QR-код, распознавание лиц, программа, безопасность, программирование.*

Abstract: *the article considers a ready-made system for identifying a person using programming languages: Python, Kotlin, SQL.*

Keywords: *QR code, face recognition, program, security, programming.*

На сегодняшний день в большинстве организаций существует система пропусков, позволяющая определить, является ли человек работником данной организации, и имеет ли он доступ в определённые секции здания. Эта система зарекомендовала себя и показала свои достоинства и недостатки. Главный недостаток этой системы – это необходимость постоянного наличия пропуска с собой. Если пропуск утерян или забыт дома приходится обращаться к

знакомым, находящихся в организации, встретить вас и подтвердить вашу личность, или вовсе ехать домой. Это может вызвать дополнительные неудобства или лишние проблемы. Однако с каждым днём информационные технологии всё интенсивно входят в нашу жизнь, и использование физических пропусков уходит в прошлое. Одним из способов отказаться от физических пропусков является создание системы проверяющей QR-код работника и идентифицирующей человека по лицу.

В данной статье рассматривается одно из возможных решений проблем, связанных с пропускными пунктами. Метод, написанный в данной работе, показывает механизм реализации системы проходного пункта в организации. Данная система позволяет избавиться от физических пропусков и сделать идентификацию человека более точной и безопасной для самой организации. Преимущества безопасности этой системы в том, что подделка данных будет практически невозможна из-за их хранения на серверах организации.

Для проектирования системы были выбраны три языка программирования. Kotlin – не только современный язык программирования, но и обратно совместим с более известным языком программирования Java. Этот язык отлично подходит для написания не только мобильного приложения под ОС Android, но и под IOS. Сервер для этой системы был написан на языке программирования Python. Этот выбор оправдан тем, что Python отлично работает с искусственным интеллектом, а он, в своё время, необходим для идентификации человека по лицу.

Рассмотрим архитектуру системы (рисунок 1).

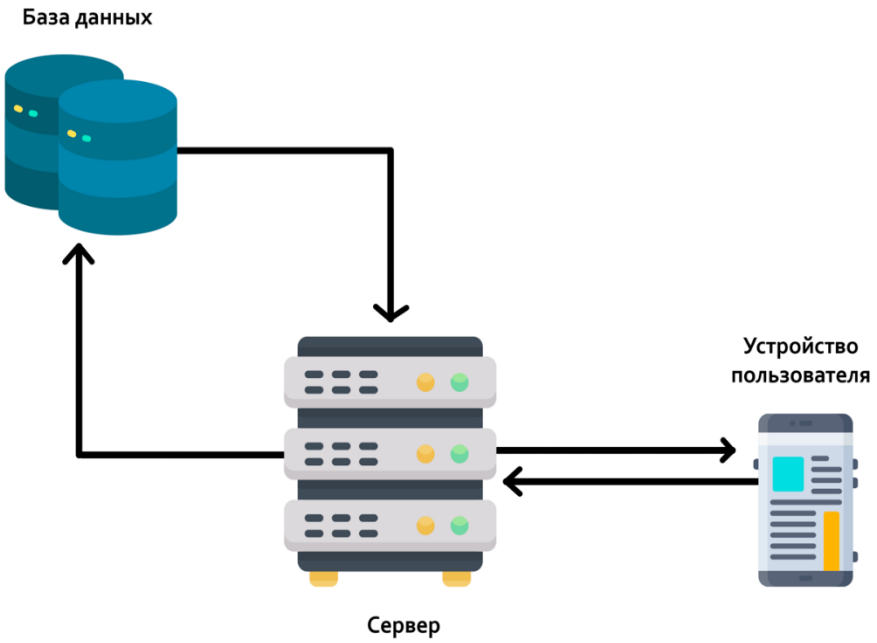

Рисунок 1. Архитектура системы

На сервере будут производиться все самые важные операции и вычисления. Это позволит снизить нагрузку на устройство пользователя и сделать идентификацию человека более конфиденциальной. Помимо этого, сервер отвечает за генерацию личных QR-кодов[1] и отправку данных на мобильные приложения.

База данных отвечает за хранение данных пользователей: фотографий, ФИО, наименование группы и личных данных.

На рисунке 2 показана ER-модель базы данных системы.

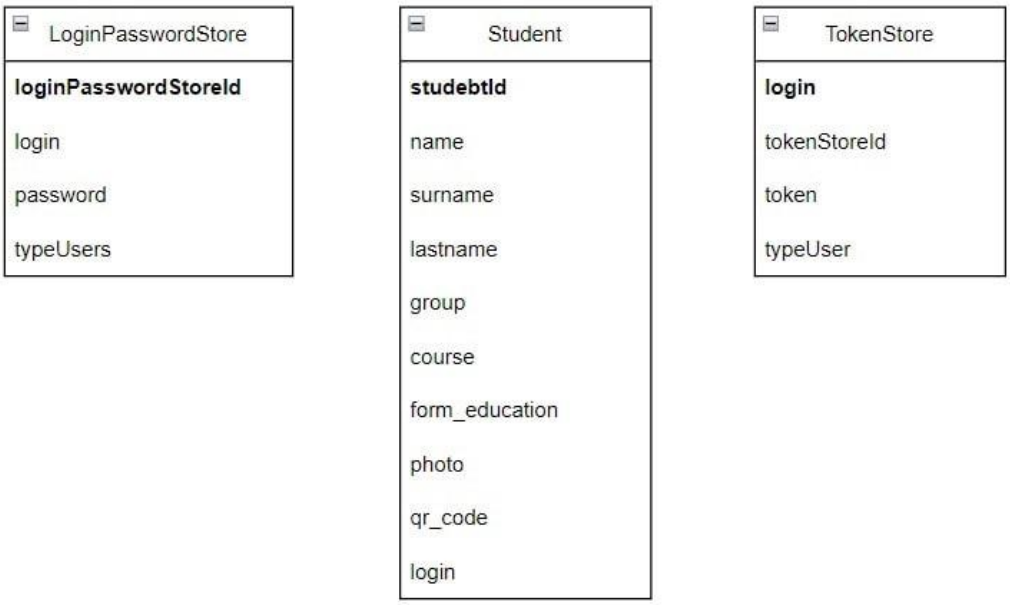

Рисунок 2. ER-модель базы данных

Мобильное приложение позволяет считывать QR-код и фотографировать человека для последующей идентификации.

При генерации QR-кода[1], сервер[2] помещает в него всю необходимую информацию о человеке: фотографию, инициалы, должность и т.п.

Программа на телефоне выполняет следующие действия:

– считывает QR-код;

– получает с него необходимые данные и отображает их;

– после считывания QR-кода, мобильное приложение запускает камеру и просит работника на пропускном пункте сфотографировать человека;

– готовый снимок отправляет на сервер[2], где происходит распознавание человека;

– выводит результата идентификации.

Итоговый цикл работы системы выглядит так:

– работника заносят в базу данных на сервер;

– сервер генерирует индивидуальный QR-код

– работник устанавливает мобильное приложение на телефон;

– работник подносит телефон к сотруднику на пропускном пункте, который считывает его QR-код;

– после считывания QR-кода работника фотографируют и происходит проверка фотографии на сервере;

– работника пропускают в организацию при успешной идентификации.

Идентификация человека по лицу может быть произведена с помощью библиотеки «Face Recognition»[3], написанной на Python и C++. Она была интегрирована в сервер[2]. Её исходный код и примеры доступны на GitHub.

На рисунке 3 показан котроллер, который отвечает за распознавание по лицу. Функция принимает два параметра: фотографию, который пользователь загружал при регистрации и фотографию, которую сделал работник пропускного пункта. Возвращаемое значение — это булева переменная, определяющая результат идентификации.

```
def face_rec(knowImage, newImage):
    known_image = face_recognition.load_image_file(knowImage)
    unknown_image = face_recognition.load_image_file(newImage)

    picture_encoding = face_recognition.face_encodings(known_image)[0]
    unknown_encoding = face_recognition.face_encodings(unknown_image)[0]

    results = face_recognition.compare_faces([picture_encoding], unknown_encoding)

    return results[0]
```

Рисунок 3. Код для идентификации пользователя по лицу

Считывание QR-кода на мобильном устройстве (рисунок 4) производится с помощью камеры и её встроенного API.

На рисунке 5 продемонстрирована генерация QR-кода по данным пользователя.

```
AndroidView(
    factory = { context ->
        val previewView = PreviewView(context)
        val preview = Preview.Builder().build()
        val selector = CameraSelector.Builder()
            .requireLensFacing(CameraSelector.LENS_FACING_BACK)
            .build()
        preview.setSurfaceProvider(previewView.surfaceProvider)
        val imageAnalysis = ImageAnalysis.Builder()
            .setTargetResolution(
                Size(
                    previewView.width,
                    previewView.height
                )
            )
            .setBackpressureStrategy(ImageAnalysis.STRATEGY_KEEP_ONLY_LATEST)
            .build()
        imageAnalysis.setAnalyzer(
            ContextCompat.getMainExecutor(context),
            QrCodeAnalyzer { result ->
                code = result
                onNewResult(result)
            }
        )
        try {
            cameraProviderFuture.get().bindToLifecycle(
                lifecycleOwner,
                selector,
                preview,
                imageAnalysis
            )
        } catch (e: Exception) {
            e.printStackTrace()
        }
        previewView ^lambda
```

Рисунок 4. Считывание QR-кода на устройстве

```
danil +1 *
def QRGenerator(url):
    QRcode = qrcode.QRCode(
        error_correction=qrcode.constants.ERROR_CORRECT_H
    )

    QRcode.add_data(url)

    QRcode.make()

    QRcolor = 'black'

    QRimg = QRcode.make_image(
        fill_color=QRcolor, back_color="white").convert('RGB')

    nameFile = url['name']

    nameQrCode = f"{nameFile}qrCod.png"
    QRimg.save(f'static/uploads/{nameQrCode}')
    return nameQrCode
```

Рисунок 5. Генерация QR-кода по данным пользователя

Использованные источники:

1. Генерация QR-кода // GitHub URL: https://github.com/lincolnloop/python-qrcode (дата обращения: 23.11.2022);

2. Web-сервер // GitHub URL: https://github.com/pallets/flask (дата обращения: 24.11.2022).

3. Распознавание по лицу // GitHub URL: https://github.com/ageitgey/face_recognition (дата обращения: 23.11.2022);