Министерство образования Новосибирской области

ГБПОУ НСО «Новосибирский авиационный технический колледж имени Б.С. Галущака»

Разработка мобильного приложения

«Личный кабинет студента НАТК»

Пояснительная записка к курсовому проекту

ПМ.01 Разработка модулей программного обеспечения для компьютерных систем

МДК.01.03 Разработка мобильных приложений

НАТКиГ.200600.010.000 ПЗ

Разработал:

Молотков Д.К.

2023

**Содержание**

[Введение 3](#_Toc135860923)

[1 Исследовательский раздел 5](#_Toc135860924)

[1.1 Описание предметной области 5](#_Toc135860925)

[1.2 Образ клиента 5](#_Toc135860926)

[1.3 Сценарии 6](#_Toc135860927)

[1.4 Сбор и анализ прототипов 7](#_Toc135860928)

[2 Проектирование приложения 12](#_Toc135860929)

[2.1 UI/UX дизайн проекта 12](#_Toc135860930)

[2.2 Выбор технологии, языка и среды программирования 17](#_Toc135860931)

[3 Разработка мобильного приложения 18](#_Toc135860932)

[3.1 Описание используемых плагинов 18](#_Toc135860933)

[3.2 Разработка базы данных 18](#_Toc135860934)

[3.3 Разработка REST API сервера 19](#_Toc135860935)

[3.4 Описание разработки мобильного приложения 23](#_Toc135860936)

[4 Тестирование 32](#_Toc135860937)

[4.1 Протокол тестирования дизайна приложения 32](#_Toc135860938)

[4.2 Протокол тестирования функционала приложения 33](#_Toc135860939)

[Заключение 35](#_Toc135860940)

[Библиография 36](#_Toc135860941)

[Приложение А 37](#_Toc135860942)

Введение

Актуальность данной темы заключается в том, что у обучающихся и преподавателей возникают проблемы с поиском и изучением расписания занятий. Мобильное приложение колледжа пригодится для того, чтобы предоставить обучающимся удобный и эффективный способ управлять своей студенческой жизнью на ходу. Это упрощает различные процессы и помогает им оставаться организованными, информированными и выполнять свои академические обязанности. Заходя в приложение, ученик сразу увидит информацию о занятиях (дата, время, кабинет, номер занятия, преподаватель). Также учащимся будет доступна страница с зачетной книжкой и личными достижениями. Преподаватель же сможет посмотреть расписание, как свое, так и по кабинетам.

Преподавателям доступна другая сторона приложения. Можно посмотреть расписание для конкретной аудитории и узнать какие там будут занятия, ознакомиться с личным расписанием занятий, в каких группах и кабинетах они будут проходить.

Целью курсового проекта является создание мобильного приложения «личный кабинет для учебного учреждения».

Задачами курсового проекта в связи с указанной целью являются:

* изучение предметной области;
* анализ прототипов приложений;
* определение сценариев использования приложения;
* спроектировать приложение;
* разработать приложение;
* протестировать приложение.

Объект исследования – приложение личный кабинет для учебного заведения.

Предмет исследования – изучение принципов функционирования и инструментов мобильного приложения для колледжа.

**Обозначения и сокращения**

JWT – JSON Web Token

ORM – Object-Relational Mapping

СУБД – Система управления базами данных

НАТК – Новосибирский авиационный технический колледж

API – Application Programming Interface

APK – Android Package Kit

HTTP – HyperText Transfer Protocol

# Исследовательский раздел

## Описание предметной области

Неотъемлемой частью учебного процесса является расписание занятий.

Очень важно иметь возможность ознакомиться с расписанием и подготовиться к занятиям.

Информацию о номере, времени занятия, преподавателе и изменениях в расписании можно узнать на стенде в учебном заведении. Так же там размещена информация об учебном процессе (учебный план), мероприятиях, выходных и праздничных днях и другая важная информация, касающаяся учебного процесса. Пользоваться только этим способом неудобно, так как будучи дома человек не сможет узнать расписание или об изменениях в нем. Есть другая возможность узнать о занятиях. Сайт учебного заведения предоставляет такую возможность. Чтобы посмотреть расписание определенной группы нужно зайти на сайт, найти раздел с расписанием и выбрать нужную группу из списка. Так же здесь можно посмотреть какие занятия и во сколько есть у конкретного преподавателя. Можно посмотреть информацию о самом преподавателе: его образование, квалификацию, фотографию, стаж и так далее. Многие участники учебного процесса используют именно сайт для поиска актуальной информации. Наиболее полезным вариантом получения информации является сайт, так как она более структурирована, а значит легче для восприятия.

## Образ клиента

Приложение предусмотрено для студентов и преподавателей Новосибирского авиационного технического колледжа имени Б.С. Галущака.

Студентами колледжа в среднем являются люди в возрасте от 15 до 25 лет. Большая часть этой аудитории пользуется смартфонами и имеет возможность пользоваться приложением. Преподавателям колледжа в среднем от 24 до 60 лет. Большинство преподавателей старшего возраста не пользуются смартфонами, и соответственно они не смогут пользоваться приложением.

## Сценарии

Целью, которую должен достигнуть пользователь, является эффективность и простота использования приложения.

Человека зачисляют в учебное заведение и ему необходимо каждый день узнавать какие занятия стоят в расписании на день, ведь расписание может меняться каждый день. Первым делом обучающийся зайдет на веб–сайт колледжа и будет искать информацию о расписании там, однако ему не обязательно искать раздел с расписанием, свою группу среди многих других. Достаточно будет скачать приложение и узнать всю актуальную информацию о занятиях там. А узнать он о приложении может на главной странице колледжа или от преподавателей. Другой вариант: студент не пользуется сайтами и смотрит расписание на стенде. Там же он может увидеть рекламу продукта и скачать его.

Закончилась академическая зачетная неделя, обучающийся на время каникул уехал к родителям и оставил зачетную книжку в колледже. Родителям хочется узнать оценки обучающегося, но ни фотографии, ни зачетной книжки под рукой нет, в отличие от телефона, на котором установлено приложение с доступом к оценкам.

Необходимо рассмотреть вариант, что потенциальным пользователем является преподаватель. Ему нужно найти аудиторию для занятий в связи с неожиданными изменениями в расписании. Чтобы это сделать, он подойдет к стенду с расписанием и будет искать свободную аудиторию для занятий. Но у него есть приложение, о котором рассказали всем преподавателям и раздали логины и пароли. Теперь ему доступно расписание по кабинетам, и он быстро найдет аудиторию.

Куратору нужно лично сообщить важную информацию своим подопечным, для этого ему нужно узнать есть ли в это время занятия у группы и если да, то в каком кабинете сейчас идут занятия. Намного удобнее и быстрее посмотреть эту информацию в смартфоне, чем идти до стенда с расписанием.

Закончилась пара, преподавателю нужно посмотреть, где и с какой группой будет проходить следующее занятие, чтобы подготовиться к нему. Вместо «веб-серфинга» сайта ему можно посмотреть расписание в приложении.

## Сбор и анализ прототипов

У данного мобильного приложения мало конкурентов, так как аудитория относительно небольшая. В частности, в магазине Play Market найдено одно подобное приложение. Рассмотрим дизайн, достоинства и недостатки приложения, имеющегося в Google Play– Расписание НАТК.

«Расписание НАТК» имеет пастельное цветовое решение. Приложение выполнено в бело–голубых тонах. Ниже приведен рисунок (рисунок 1) с изображением главной, страницы с расписанием.

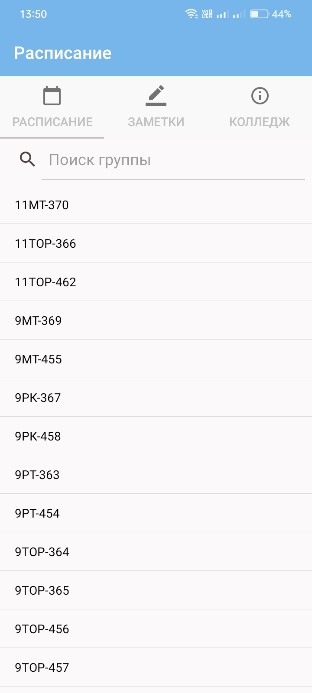


Рисунок 1 – Главная страница, страница с расписанием

В приложении есть 3 раздела: расписание, заметки и колледж. В первом разделе содержится список всех групп. При выборе группы открывается окно с выбором дня недели и соответствующей даты (рисунок 2).

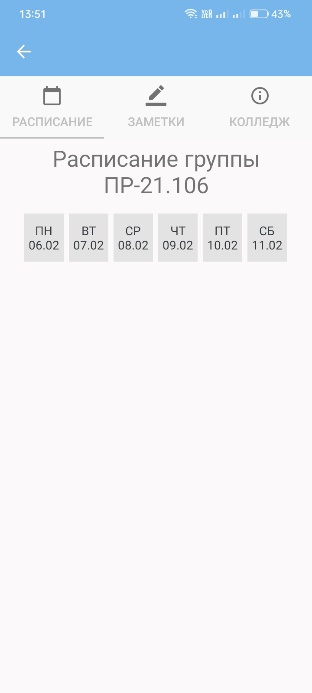


Рисунок 2 – страница группы с выбором дня

После выбора дня недели открывается окно с расписанием   
(рисунок 3), на котором представлено расписание для конкретного дня.



Рисунок 3 – страница с расписанием группы на день

Второй раздел приложения – «заметки». При выборе раздела открывается пустая страница с возможностью создания напоминания или другой заметки (рисунок 4).



Рисунок 4 – страница «заметки»

После нажатия на «+» появляется поле, в котором можно записать заметку (рисунок 5).



Рисунок 5 – страница с полем для заметок

В третьем разделе «колледж» находится краткая информация о приложении, полное название колледжа с логотипом и действующий ссылки на сайт колледжа и разработчика (рисунок 6).



Рисунок 6 – страница «колледж»

В итоге проделанной работы можно сделать выводы о приложении.

Плюсы:

* нет перегрузки пространства деталями;
* наличие вкладки «заметки». Можно записывать домашние задания.

Недостатки:

* бледный дизайн, много пустого пространства;
* маленький «+» и плохо заметный на вкладке заметки;
* много действий чтобы посмотреть расписание на день (найти группу, выбрать день);
* невозможность посмотреть свои оценки за семестр;
* нет интерфейса для преподавателей;
* при выборе групп «11 ТОР-462», «11 ТОР-366», «9 РТ-458» приложение аварийно завершает свою работу;
* не предусмотрена «темная тема».

В заключении актуальными задачами разработки являются: создание приятного, но не броского дизайна, создание интерфейса для преподавателей, возможность просмотра оценок обучающимся, быстрый доступ к расписанию своей группы.

# Проектирование приложения

## UI/UX дизайн проекта

Для разработки дизайна проекта выбрана программа Figma

Для проекта были определены основные экраны:

* авторизации;
* просмотра расписания;
* выбора режима просмотра расписания
* личного кабинета.

Цветовая гамма приложения взята на основе цветов логотипа учебного заведения.

Ниже на рисунке 7 представлен логотип учебного заведения.

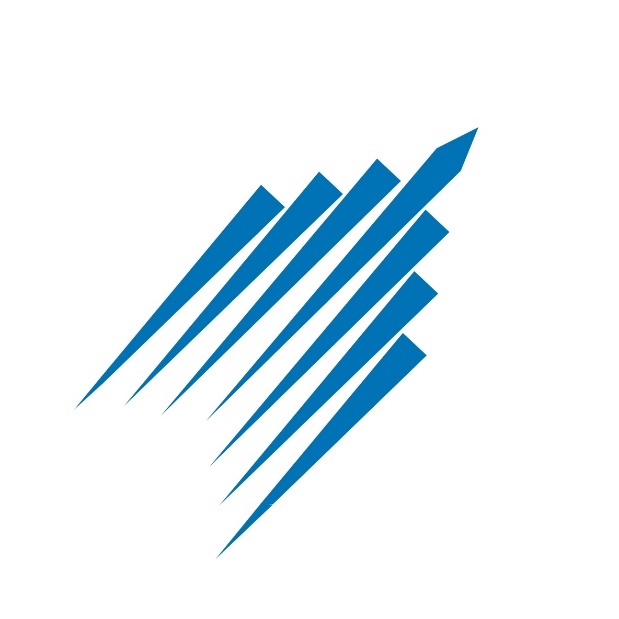


Рисунок 7 – Логотип учебного заведения

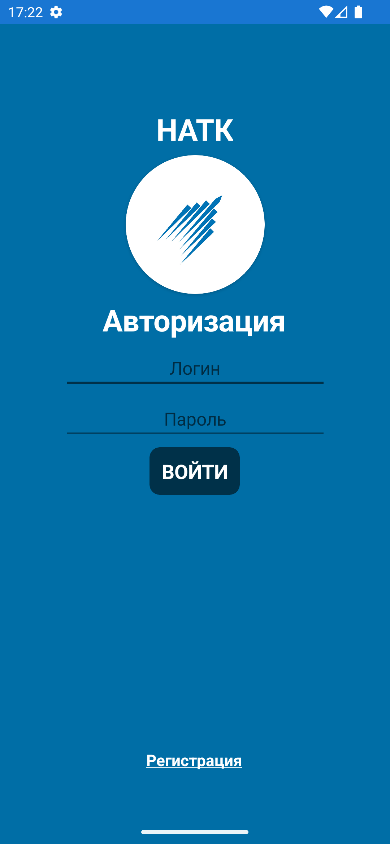
На рисунке 8 ниже представлена страница входа в приложение. 

Рисунок 8 – Страница «Входа»

На данной странице имеется возможность входа в приложение, если пользователь ещё не зарегистрирован, он может перейти к отправке запроса на регистрацию.

На рисунке 9 ниже представлена страница просмотра расписания. 

Рисунок 9 – Страница «Просмотр расписания»

На данной странице имеется возможность просмотра расписания. На странице отражены даты и дни недели, наименование дисциплины, номер пары, время пары, ФИО преподавателя и аудитория. Так же отмечается информация о выходных днях.

На рисунке 10 ниже представлена страница Выбора режима просмотра расписания.

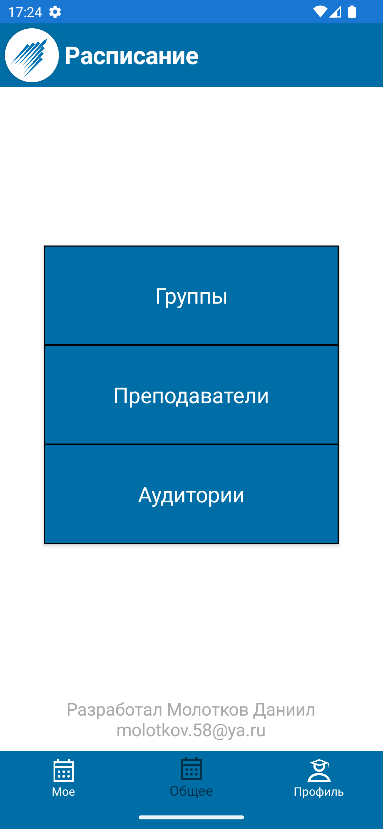


Рисунок 10 – Страница «Выбора режима просмотра расписания»

На данной странице имеется возможность выбора режима просмотра расписания. Доступны следующие режимы: группы, преподаватели и аудитории.

На рисунке 11 ниже представлена страница Выбора режима просмотра расписания.



Рисунок 11 – Страница «Просмотр расписания в режиме группы»

На данной странице имеется возможность выбора группы и просмотра расписания для нее, отражены даты и дни недели, номер группы, наименование дисциплины, номер пары, время пары, ФИО преподавателя и аудитория. Так же отмечается информация о выходных днях.

На рисунке 12 ниже представлена страница Выбора режима просмотра расписания.



Рисунок 12 – Страница «Просмотр расписания в режиме преподаватели»

На данной странице имеется возможность выбора преподавателя и просмотра расписания для него, отражены даты и дни недели, номер группы, наименование дисциплины, номер пары, время пары, ФИО преподавателя и аудитория. Так же отмечается информация о выходных днях.

## Выбор технологии, языка и среды программирования

В качестве основного языка программирования для разработки мобильного приложения выступит C# с фреймворком Xamarin.Forms. Это позволит получить кроссплатформенное приложение, как для Android, так и для IOS.

Базой данных выступит MariaDB. PhpMyAdmin позволит легко управлять базой данных во время разработки приложения.

Для разработки сервера REST API используется язык программирования Python и фреймворк FastAPI и ORM SQLalchemy. Это позволит в короткие сроки написать быстрый и эффективный сервер для приложения. Приятным бонусом будет автоматически создаваемая документация к серверу.

# Разработка мобильного приложения

## Описание используемых плагинов

Xamarin.Forms – фреймворк для кросс-платформенной разработки мобильных приложений с использованием языка программирования C#. Он позволяет создавать приложения, работающие на разных платформах (iOS, Android, Windows) с общим кодом. Xamarin.Forms предоставляет инструменты и библиотеки для создания интерфейса, взаимодействия с устройствами и разработки логики приложения. Он обеспечивает доступ к возможностям каждой платформы и интегрируется с популярными инструментами разработки.

Фреймворк FastAPI: FastAPI – это быстрый (high-performance) веб-фреймворк для разработки API на языке Python. Он основан на стандарте OpenAPI и предоставляет автоматическую генерацию документации, асинхронную обработку запросов, валидацию данных и другие возможности для удобной разработки API.

ORM SQLAlchemy: SQLAlchemy – это инструмент для языка программирования Python. Он позволяет разработчикам работать с базами данных, используя объектно-ориентированный подход, а не прямые SQL-запросы. SQLAlchemy обеспечивает удобные методы для взаимодействия с базой данных, выполнения запросов и управления данными.

База данных MariaDB: MariaDB – это реляционная база данных, являющаяся форком MySQL. Она предлагает совместимость с MySQL, но также включает в себя дополнительные функции и улучшения производительности. MariaDB обеспечивает надежное хранение данных и поддерживает широкий набор операций с базой данных.

## Разработка базы данных

Используя СУБД MariaDB создана база данных, сущности которой определены в ходе анализа предметной области. Структура отражена в ER-диаграмме, которая представлена на рисунке 13. Сущность «Users» хранит информации о зарегистрированных пользователях приложения, «Roles» хранит информацию о ролях, «Groups» хранит группы, «SubGroups» хранит подгруппы.

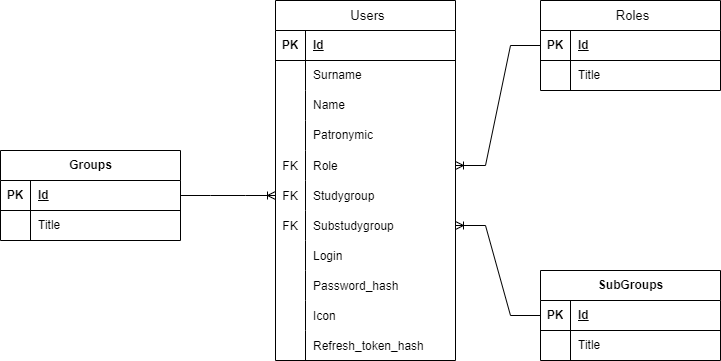


Рисунок 13 – ER диаграмма базы данных

## Разработка REST API сервера

Без серверной части невозможно начать разработку мобильного приложения.

С помощью команды «pip install FastAPI[ALL]» установлен фреймворк для разработки веб-приложений и API на языке Python – FastAPI.

Далее используя ORM SQLalchemy созданы модели сущностей базы данных, которые приведены на листинге 1.

Листинг 1 – модели сущностей базы данных

from sqlalchemy import Column, Integer, String, Float, SmallInteger, ForeignKey

from database import Base, engine

class User(Base):

    \_\_tablename\_\_ = "Users"

    Id = Column(Integer, primary\_key = True, index = True)

    Surname = Column(String(length=150), nullable=False)

    Name = Column(String(length=150), nullable=False)

    Patronymic = Column(String(length=150), nullable=True)

    Role = Column(Integer, ForeignKey("Roles.Id"))

    Studygroup = Column(Integer, ForeignKey("Groups.Id"))

    Substudygroup = Column(Integer, ForeignKey("SubGroups.Id"))

    Login = Column(String(length=50), nullable=False)

    Password\_hash = Column(String(length=255), nullable=False)

    Icon = Column(String(length=150), nullable=True)

    Refresh\_token\_hash = Column(String(length=512), nullable=True)

class Role(Base):

    \_\_tablename\_\_ = "Roles"

    Id = Column(Integer, primary\_key = True, index = True)

    Name = Column(String(length=150), nullable=False)

class Group(Base):

    \_\_tablename\_\_ = "Groups"

    Id = Column(Integer, primary\_key = True, index = True)

    Name = Column(String(length=50), nullable=False)

class SubGroup(Base):

    \_\_tablename\_\_ = "SubGroups"

    Id = Column(Integer, primary\_key = True, index = True)

    Name = Column(SmallInteger, nullable=False)

class Image(Base):

    \_\_tablename\_\_ = 'Images'

    id = Column(Integer, primary\_key=True, index=True)

    file\_id = Column(Integer)

    name = Column(String(length=150))

    tag = Column(String(length=150))

    size = Column(Integer)

    mime\_type = Column(String(length=150))

    modification\_time = Column(String(length=150))

Base.metadata.create\_all(engine)

Написан функционал обработки запросов к серверу, выбора данных, проверки данных и формирования ответов на запросы. Для каждого запроса создается маршрут, функция обработчик и сервис.

Маршрут позволяет обращаться к серверу через протокол HTTP с помощью get, post, put и других запросов. Для этого пишется декоратор с типом HTTP запроса и внутри указывается путь к этому запросу. В итоге, путь к запросу формируется следующим образом «http://<ip адрес сервера или домен>:порт/строка, указанная в декораторе».

Функция обработчик вызывается при поступлении запроса на маршрут. В ней указана логика обработки данных запроса, обращения к сервисам и возвращения ответа. Ниже приведен фрагмент кода с маршрутом и функцией обработчиком авторизации на листинге 2.

Листинг 2 – фрагмент кода маршрутов и функций обработчиков авторизации

@app.post('/auth/signup')//обработка запроса на регистрацию

def signup(user: schemas.UserCreate, db: Session = Depends(get\_db)):

    if getUser(db,user.Login) != None:

        return 'Account already exists'

    hashed\_password = auth\_handler.encode\_password(user.Password)

    user.Password = hashed\_password

    return postUser(db,user)

@app.post('/api/auth/login', tags=["Auth"])//обработка запроса на авторизацию

def login(user\_details: schemas.UserLogin, db: Session = Depends(get\_db)):

    user = getUser(db,user\_details.Login)

    if (user is None):

        raise HTTPException(status\_code=401, detail='Invalid username')

    if (not auth\_handler.verify\_password(user\_details.Password, user.Password\_hash)):

        raise HTTPException(status\_code=401, detail='Invalid password')

    access\_token = auth\_handler.encode\_token(user.Login)

    refresh\_token = auth\_handler.encode\_refresh\_token(user.Login)

    updateUserRefreshToken(db, user, auth\_handler.get\_token\_hash(refresh\_token))

    return {'AccessToken': access\_token, 'RefreshToken': refresh\_token}

@app.post('/api/auth/refresh', tags=["Auth"])//обработка запроса на обновление токенов

def refresh\_token(credentials: HTTPAuthorizationCredentials = Security(security), db: Session = Depends(get\_db)):

    refresh\_token = credentials.credentials

    user = getUser(db, auth\_handler.decode\_refresh\_token(refresh\_token))

    if(not auth\_handler.verify\_Tokens(refresh\_token, user.Refresh\_token\_hash)):

        updateUserRefreshToken(db, user, None)

        raise HTTPException(status\_code=403, detail = 'Invalid refresh token')

    new\_token = auth\_handler.refresh\_token(refresh\_token)

    refresh\_token = auth\_handler.encode\_refresh\_token(user.Login)

    updateUserRefreshToken(db, user, auth\_handler.get\_token\_hash(refresh\_token))

    return {'AccessToken': new\_token, 'RefreshToken': refresh\_token}

Сервисы нужны для получения информации из базы данных, проверки вводной информации, создания новых объектов в базе данных и т.п. Ниже приведен фрагмент кода сервиса авторизации на листинге 3.

Листинг 3 – фрагмент кода сервиса авторизации

def getUser(db: Session, login: str): #получение пользователя из базы данных

    return db.query(models.User).filter(models.User.Login == login).first()

def postUser(db: Session, User: UserCreate): #запись пользователя

    db\_user = models.User(

        Surname=User.Surname,

        Name=User.Name,

        Patronymic=User.Patronymic,

        Studygroup=User.Studygroup,

        Substudygroup=User.Substudygroup,

        Login=User.Login,

        Password\_hash=User.Password,

        Icon=User.Icon,

        Role=User.Role

    )

    if db\_user == None:

        raise HTTPException(404)

    db.add(db\_user)

    db.commit()

    return db\_user

def updateUserRefreshToken(db: Session, user: models.User, refresh\_token\_hash): #обновление токена в базе данных

    i = db.query(models.User).get(user.Id)

    i.Refresh\_token\_hash = refresh\_token\_hash

    db.add(i)

    db.commit()

def toUserOut(db: Session, user: models.User): #преобразование объекта пользователя к формату выходных данных

    userOut = schemas.UserOut(

        Surname=user.Surname,

        Name=user.Name,

        Patronymic=user.Patronymic,

        Studygroup=db.query(models.Group).filter(

            models.Group.Id == user.Studygroup).first().Name,

        Substudygroup=user.Substudygroup,

        Role=user.Role,

        Login=user.Login

    )

    return userOut

В разработке используется авторизация с помощью JWT токенов. Токен состоит из заголовка, полезной нагрузки (определяемая разработчиком) и контрольной суммы, а также имеет ограниченный срок жизни. Пользователь получает два токена: для авторизации и обновления. Хэш токена обновления записывается в базу данных для безопасности. Если токен авторизации украден, он будет действителен только ограниченное время. Если украден токен обновления, злоумышленник не сможет создать новые токены после того, как пользователь авторизуется снова. Все запросы защищены и требуют токена авторизации, за исключением запросов на авторизацию.

## Описание разработки мобильного приложения

Для разработки мобильного приложения выбран фреймворк Xamarin.Forms.

C помощью средств этого фреймворка разработаны экраны приложения в соответствии с дизайном. Xamarin.Forms позволяет создавать экраны приложения с использованием языка разметки xaml. Ниже на листинге 4 приведен экран просмотра расписания.

Листинг 4 – код дизайна экрана просмотра расписания

<?xml version="1.0" encoding="utf-8" ?>

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

             xmlns:view="clr-namespace:Course\_appForms.View;assembly=Course\_appForms"

             x:Class="Course\_appForms.View.Shedule"

             Shell.NavBarIsVisible="False">

    <StackLayout BackgroundColor="{AppThemeBinding Light=#fff, Dark=#171717}">

        <StackLayout Orientation="Horizontal" BackgroundColor="{DynamicResource Main}" Padding="5">

            <Frame CornerRadius="150" WidthRequest="15" BackgroundColor="White">

                <Image Source="LOGO\_NATK.jpg" Aspect="AspectFill" Scale="3.8"/>

            </Frame>

            <Label Text="Расписание" HorizontalTextAlignment="End" VerticalOptions="CenterAndExpand" HorizontalOptions="Start" FontSize="25" FontAttributes="Bold" TextColor="White"/>

            <Label x:Name="lbGroup" Text="" HorizontalTextAlignment="End" HorizontalOptions="EndAndExpand" VerticalOptions="Center" TextColor="White" FontAttributes="Bold" Padding="0,0, 5, 0" />

        </StackLayout>

        <Label x:Name="lbInfo" FontSize="20" Text="" IsVisible="False" HorizontalTextAlignment="Center" HorizontalOptions="CenterAndExpand" VerticalOptions="Center" TextColor="{AppThemeBinding Light=#000, Dark=#fff}" FontAttributes="Bold" Padding="5,0, 5, 0" />

        <RefreshView x:Name="refreshView" BackgroundColor="{AppThemeBinding Light=#fff, Dark=#949494}">

            <CollectionView Margin="20, 0,20,5" x:Name="CvShedule" HorizontalOptions="Start" BackgroundColor="{AppThemeBinding Light=#fff, Dark=#171717}">

                <CollectionView.ItemsLayout>

                    <LinearItemsLayout Orientation="Vertical" ItemSpacing="10" >

                    </LinearItemsLayout>

                </CollectionView.ItemsLayout>

                <CollectionView.ItemTemplate>

                    <DataTemplate>

                        <StackLayout >

                            <Frame CornerRadius="0" BorderColor="{AppThemeBinding Light=#000, Dark=#fff}" Padding="1,1,1,1" Margin="0" BackgroundColor="{AppThemeBinding Light=#fff, Dark=#949494}">

                                <StackLayout BackgroundColor="{AppThemeBinding Light=#fff, Dark=#949494}">

                                    <Frame BackgroundColor="{DynamicResource Main}"

                                       Padding="0"

                                       HeightRequest="40"

                                       CornerRadius="0"

                                       BorderColor="{AppThemeBinding Light=#000, Dark=#fff}"

                                       Margin="0,0,0,-11"

                                       VerticalOptions="StartAndExpand">

                                        <Label Text="{Binding Date}"

                                           TextColor="White"

                                           FontAttributes="Bold"

                                           HorizontalOptions="Center"

                                           VerticalOptions="CenterAndExpand"

                                           FontSize="16"

                                           Padding="0"/>

                                    </Frame>

                                    <ScrollView Orientation="Vertical" >

                                        <StackLayout Orientation="Vertical" Spacing="0" Padding="0,0,0,0"

                                                 BindableLayout.ItemsSource="{Binding listOfLessons}" BindableLayout.ItemTemplateSelector="{StaticResource sheduleDataTemplateSelector}">

                                        </StackLayout>

                                    </ScrollView>

                                </StackLayout>

                            </Frame>

                        </StackLayout>

                    </DataTemplate>

                </CollectionView.ItemTemplate>

            </CollectionView>

        </RefreshView>

    </StackLayout>

    <ContentPage.Resources>

        <ResourceDictionary>

            <DataTemplate x:Key="StudentTemplate">

                <Frame Padding="0" BorderColor="{AppThemeBinding Light=#000, Dark=#fff}" CornerRadius="0" BackgroundColor="{AppThemeBinding Light=#fff, Dark=#525252}" Margin="0">

                    <Grid ColumnDefinitions="0.18\*,0.82\*" >

                        <StackLayout Grid.Column="0"  Padding="5,0,0,0" >

                            <Label Text="{Binding GetNumber}"

                                   TextColor="{AppThemeBinding Light=#000, Dark=#fff}"

                                                               FontSize="16"

                                                               VerticalOptions="StartAndExpand"

                                                               Padding="5,5,0,0"/>

                            <Label Text="{Binding Time}"

                                   TextColor="{AppThemeBinding Light=#000, Dark=#fff}"

                                                               FontAttributes="Bold"

                                                               FontSize="16"

                                                               VerticalOptions="EndAndExpand"

                                                               Padding="5,15,0,5"/>

                        </StackLayout>

                        <StackLayout Grid.Column="1" HorizontalOptions="EndAndExpand"  Padding="0,5,10,5">

                            <Label Text="{Binding Title}"

                                   TextColor="{AppThemeBinding Light=#000, Dark=#fff}"

                                                                       FontAttributes="Bold"

                                                                       VerticalOptions="StartAndExpand"

                                                                       HorizontalTextAlignment="End"

                                                                       Padding="0,0,0,2"/>

                            <view:IdLable LbId="0" Text="{Binding Teacher}"

                                          TextColor="{AppThemeBinding Light=#000, Dark=#fff}"

                                          TextDecorations="Underline"

                                          VerticalOptions="CenterAndExpand"

                                          HorizontalTextAlignment="End"

                                          Padding="0,0,0,2">

                                <view:IdLable.GestureRecognizers>

                                    <TapGestureRecognizer Tapped="TapGestureRecognizer\_OnTapped"/>

                                </view:IdLable.GestureRecognizers>

                            </view:IdLable>

                            <Label Text="{Binding ClassRoom}"

                                                                       TextColor="{AppThemeBinding Light=#000, Dark=#fff}"

                                                                       VerticalOptions="EndAndExpand"

                                                                       HorizontalTextAlignment="End"

                                                                       Padding="0,0,0,1"/>

                        </StackLayout>

                    </Grid>

                </Frame>

            </DataTemplate>

            <DataTemplate x:Key="CenterLessonStudentTemplate">

                <Frame Padding="1" BorderColor="{AppThemeBinding Light=#000, Dark=#fff}" CornerRadius="0" BackgroundColor="{AppThemeBinding Light=#fff, Dark=#525252}" Margin="0">

                    <StackLayout HorizontalOptions="EndAndExpand"  Padding="0,5,0,5">

                        <Label Text="{Binding Title}"

                               TextColor="{AppThemeBinding Light=#000, Dark=#fff}"

                               HorizontalTextAlignment="Center"

                                                                       FontAttributes="Bold"

                                                                       VerticalOptions="CenterAndExpand"

                                                                       Padding="0,0,0,1"/>

                    </StackLayout>

                </Frame>

            </DataTemplate>

            <DataTemplate x:Key="TeacherTemplate">

                <Frame Padding="0" BorderColor="{AppThemeBinding Light=#000, Dark=#fff}" BackgroundColor="{AppThemeBinding Light=#fff, Dark=#525252}" CornerRadius="0">

                    <Grid ColumnDefinitions="0.18\*,0.82\*" >

                        <StackLayout Grid.Column="0"  Padding="5,0,0,0">

                            <Label Text="{Binding GetNumber}"

                                                               TextColor="{AppThemeBinding Light=#000, Dark=#fff}"

                                                               FontSize="18"

                                                               VerticalOptions="StartAndExpand"

                                                               Padding="5,5,0,0"/>

                            <Label Text="{Binding Time}"

                                                               TextColor="{AppThemeBinding Light=#000, Dark=#fff}"

                                                               FontAttributes="Bold"

                                                               FontSize="16"

                                                               VerticalOptions="EndAndExpand"

                                                               Padding="5,15,0,5"/>

                        </StackLayout>

                        <StackLayout Grid.Column="1" HorizontalOptions="EndAndExpand"  Padding="0,5,10,5">

                            <Label Text="{Binding Title}"

                                                               TextColor="{AppThemeBinding Light=#000, Dark=#fff}"

                                                               FontAttributes="Bold"

                                                               VerticalOptions="StartAndExpand"

                                                               HorizontalTextAlignment="End"

                                                               Padding="0,0,0,2"/>

                            <view:IdLable LbId="1" Text="{Binding Group}"

                                          TextDecorations="Underline"

                                                               TextColor="{AppThemeBinding Light=#000, Dark=#fff}"

                                                               VerticalOptions="CenterAndExpand"

                                                               HorizontalTextAlignment="End"

                                                               Padding="0,0,0,2"

                                                               >

                                <view:IdLable.GestureRecognizers>

                                    <TapGestureRecognizer Tapped="TapGestureRecognizer\_OnTapped"/>

                                </view:IdLable.GestureRecognizers>

                            </view:IdLable>

                            <Label Text="{Binding ClassRoom}"

                                                               TextColor="{AppThemeBinding Light=#000, Dark=#fff}"

                                                               VerticalOptions="EndAndExpand"

                                                               HorizontalTextAlignment="End"

                                                               Padding="0,0,0,1"/>

                        </StackLayout>

                    </Grid>

                </Frame>

            </DataTemplate>

            <view:SheduleDataTemplateSelector x:Key="sheduleDataTemplateSelector"

                                              StudentTemplate="{StaticResource StudentTemplate}"

                                              TeacherTemplate="{StaticResource TeacherTemplate}"

                                              CenterLessonStudentTemplate="{StaticResource CenterLessonStudentTemplate}" />

        </ResourceDictionary>

    </ContentPage.Resources>

</ContentPage>

После того как верстка для всех экранов закончена, можно приступить к функциональной части приложения. Созданы методы для выполнения запросов с использованием токенов. Написаны модели, которые клиент получает от сервера. И после все необходимые методы для каждого возможного события. Ниже на листинге 5 приведен код функционала экрана просмотра расписания.

Листинг 5 – функционал экрана просмотра расписания

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Input;

using Course\_appForms.Models;

using Course\_appForms.Services;

using Xamarin.CommunityToolkit.Extensions;

using Xamarin.Essentials;

using Xamarin.Forms;

using Xamarin.Forms.Xaml;

using Newtonsoft.Json;

namespace Course\_appForms.View

{

    [XamlCompilation(XamlCompilationOptions.Compile)]

    public partial class Shedule : ContentPage

    {

        private User \_user;

        private string \_group;

        private string \_teacher;

        private bool \_offline;

        public Shedule()//конструктор класса, загрузка данных из памяти

        {

            InitializeComponent ();

            if (Connectivity.NetworkAccess != NetworkAccess.Internet)

            {

                if (!App.Database.GetItems().Any())

                {

                    this.DisplayToastAsync("Нет подключения к сети", 2000);

                    return;

                }

                \_user = (User)Application.Current.Properties["user"];

                var list = JsonConvert.DeserializeObject<List<Models.Shedule>>(App.Database.GetItems().FirstOrDefault().SheduleDamp);

                Device.BeginInvokeOnMainThread(() => CvShedule.ItemsSource = list);

                if (\_user.Role != 2) lbGroup.Text = \_user.Studygroup;

                \_offline = true;

                refreshView.Command = new RefreshCommand();

                refreshView.CommandParameter = this;

                this.DisplayToastAsync("Загруженные данные", 2000);

            }

            else

            {

                \_user = (User)Application.Current.Properties["user"];

                Load();

                refreshView.Command = new RefreshCommand();

                refreshView.CommandParameter = this;

                refreshView.IsRefreshing = true;

            }

        }

        public Shedule(string Group) //конструктор класса, загрузка данных из памяти

        {

            InitializeComponent();

            if (Connectivity.NetworkAccess != NetworkAccess.Internet)

            {

                this.DisplayToastAsync("Нет подключения к сети", 5000);

                return;

            }

            \_user = (User)Application.Current.Properties["user"];

            \_group = Group;

            Load();

            refreshView.Command = new RefreshCommand();

            refreshView.CommandParameter = this;

            refreshView.IsRefreshing = true;

        }

        public Shedule(string Teacher, bool temp = false) //конструктор класса, загрузка данных из памяти

        {

            InitializeComponent();

            if (Connectivity.NetworkAccess != NetworkAccess.Internet)

            {

                this.DisplayToastAsync("Нет подключения к сети", 5000);

                return;

            }

            \_user = (User)Application.Current.Properties["user"];

            \_teacher = Teacher;

            Load();

            refreshView.Command = new RefreshCommand();

            refreshView.CommandParameter = this;

            refreshView.IsRefreshing = true;

        }

        public async void Refesh()//обновление информации на странице

        {

            if (Connectivity.NetworkAccess != NetworkAccess.Internet)

            {

                await this.DisplayToastAsync("Нет подключения к сети", 5000);

                refreshView.IsRefreshing = false;

                \_offline = true;

            }

            else

            {

                \_offline = false;

            }

            if (\_offline)

            {

                refreshView.IsRefreshing = false;

                return;

            }

            if (\_group != null)

            {

                var list = await SheduleService.GetShedule(\_group, 1);

                Device.BeginInvokeOnMainThread(() => CvShedule.ItemsSource = list);

            }

            else if (\_teacher != null)

            {

                var list = await SheduleService.GetSheduleTeacher($"{\_teacher}");

                Device.BeginInvokeOnMainThread(() => CvShedule.ItemsSource = list);

            }

            else if (\_user.Role == 1)

            {

                var list = await SheduleService.GetShedule(\_user.Studygroup, \_user.Substudygroup);

                Device.BeginInvokeOnMainThread(() => CvShedule.ItemsSource = list);

                App.Database.TruncateTable();

                App.Database.SaveItem(new SheduleDB {SheduleDamp = JsonConvert.SerializeObject(list)});

            }

            else if (\_user.Role == 2)

            {

                var list = await SheduleService.GetSheduleTeacher($"{\_user.Surname} {\_user.Name} {\_user.Patronymic}");

                Device.BeginInvokeOnMainThread(() => CvShedule.ItemsSource = list);

                Application.Current.Properties["Shedule"] = list;

                App.Database.TruncateTable();

                App.Database.SaveItem(new SheduleDB {SheduleDamp = JsonConvert.SerializeObject(list)});

            }

            refreshView.IsRefreshing = false;

        }

        protected void Load()//загрузка данных

        {

            if (\_group == null && \_teacher == null)

            {

                if (\_user.Role != 2) lbGroup.Text = \_user.Studygroup;

            }else if (\_group != null)

            {

                lbInfo.Text = \_group;

                lbInfo.IsVisible = true;

            }

            else if (\_teacher != null)

            {

                lbInfo.Text = \_teacher;

                lbInfo.IsVisible = true;

            }

        }

        public class RefreshCommand : ICommand//интерфейс обновления данных на странице

        {

            public bool CanExecute(object parameter)

            {

                return true;

            }

            public void Execute(object parameter)

            {

                Shedule shedule = (Shedule)parameter;

                shedule.Refesh();

            }

            public event EventHandler CanExecuteChanged

            {

                add { }

                remove { }

            }

        }

        private async void TapGestureRecognizer\_OnTapped(object sender, EventArgs e)//обработчик нажатий

        {

            if (\_group == null && \_teacher == null && \_offline == false)

            {

                var LbSender = (IdLable)sender;

                if (LbSender.LbId == 0)

                {

                    await Navigation.PushModalAsync(new Shedule(Teacher:LbSender.Text));

                }

                else if (LbSender.LbId == 1)

                {

                    await Navigation.PushModalAsync(new Shedule(Group:LbSender.Text));

                }

            }

        }

    }

    public class SheduleDataTemplateSelector : DataTemplateSelector//обработчик шаблонов отображения

    {

        public DataTemplate StudentTemplate { get; set; }

        public DataTemplate TeacherTemplate { get; set; }

        public DataTemplate CenterLessonStudentTemplate { get; set; }

        protected override DataTemplate OnSelectTemplate(object item, BindableObject container)

        {

            if (((Lesson)item).Mode == 0)

            {

                if (((Lesson)item).Number == 0) return CenterLessonStudentTemplate;

            }

            else

            {

                if (((Lesson)item).Mode == 1) return TeacherTemplate;

                return StudentTemplate;

            }

            return StudentTemplate;

        }

    }

}

После верстки и разработки функционала проект упаковывается в APK файл для установки приложения на устройство.

# Тестирование

## Протокол тестирования дизайна приложения

Для проведения тестирования дизайна приложения выбрано устройство Realme 10 Pro Plus 5G c разрешением экрана 2412×1080 пикселей, android 13.

Тест 1 – сравнение экрана «Авторизация» в макете и приложении

На рисунке 14 показано сравнение экрана с дизайном макета и скриншота экрана приложения. В результате сравнения определено, что макет практически полностью совпадает с приложением.

Проверка пройдена.

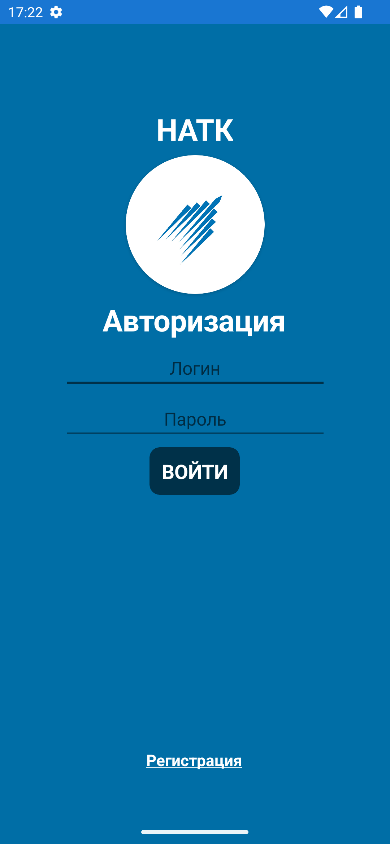
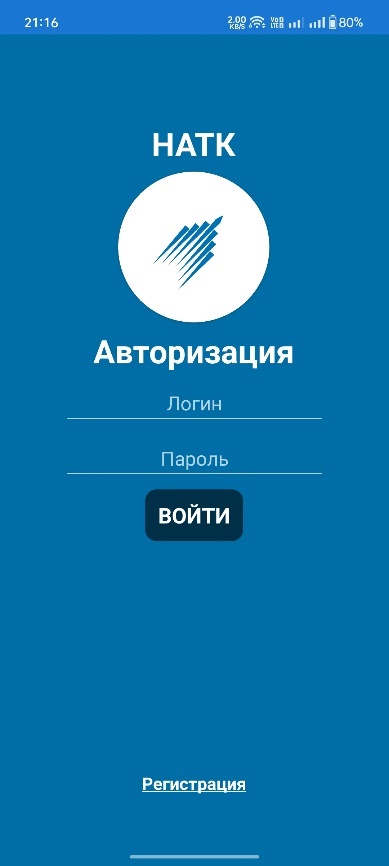
 

Рисунок 14 – сравнение экрана макета и скриншота приложения

Тест 2 – сравнение экрана «Просмотр расписания» в макете и приложении

На рисунке 15 показано сравнение экрана с дизайном макета и скриншота экрана приложения. В результате сравнения определено, что макет практически полностью совпадает с приложением.

Проверка пройдена.

Рисунок 15 – сравнение экрана макета и скриншота приложения

## Протокол тестирования функционала приложения

Название тест кейса: Просмотр расписания во вкладке «Расписание»

Предусловия тест кейса:

* Установлено и запущено мобильное приложение «Мой НАТК»;
* Пользователь авторизован в приложении.

Шаги тест кейса:

* открыть мобильное приложение «Мой НАТК»;
* перейти на вкладку «Расписание».

Ожидаемый результат тест кейса:

* отображается расписание занятий для выбранного пользователя;
* расписание отображается в удобочитаемом формате, с указанием даты, времени и аудитории;
* расписание соответствует учебному плану и актуальным данным.

Статус кейса: Пройден

Название тест кейса: Выбор режима просмотра расписания во вкладке «Общее»

Предусловия тест кейса:

* установлено и запущено мобильное приложение «Мой НАТК»;
* пользователь авторизован в приложении.

Шаги тест кейса:

* открыть мобильное приложение «Мой НАТК»;
* перейти на вкладку «Общее»;
* выбрать режим просмотра расписания;
* заполнить поля.

Ожидаемый результат тест кейса:

* расписание отображается в выбранном режиме просмотра;
* расписание соответствует актуальным данным и корректно отображается в выбранном режиме.

Статус кейса: Пройден

Название тест кейса: Просмотр информации во вкладке «Личный кабинет»

Предусловия тест кейса:

* установлено и запущено мобильное приложение «Мой НАТК»;
* пользователь авторизован в приложении.

Шаги тест кейса:

* открыть мобильное приложение «Мой НАТК»;
* перейти на вкладку «Личный кабинет».

Ожидаемый результат тест кейса:

* отображается информация о пользователе, включая фамилию и инициалы;
* отображается номер учебной группы, к которой принадлежит пользователь.

Статус кейса: Пройден

Заключение

В рамках курсового проекта разработано мобильное приложение «Мой НАТК», которое предоставляет удобный и функциональный инструмент для студентов. Приложение позволяет студентам эффективно управлять своим учебным процессом и получать необходимую информацию.

Одной из ключевых возможностей приложения является просмотр расписания, доступный через вкладку «Расписание». Студенты могут легко ознакомиться со своим расписанием занятий, включающим даты, время и аудитории. Важно отметить, что приложение обеспечивает актуальность расписания, регулярно обновляя его на основе изменений в учебном плане. Это позволяет студентам быть в курсе всех изменений и своевременно планировать свою учебную нагрузку.

В результате выполнения задач создано приложение, отвечающее требованиям предметной области, обладающее удобным пользовательским интерфейсом, успешно проходящее тестирование и демонстрирующее свою функциональность и надежность.

В качестве развития разработки, можно предложить добавление зачетной книжки, добавление портфолио, создание доступа родителям для контроля успеваемости, создание доски объявлений для кураторов и старост групп. Добавление функционала по отчетности о посещаемости группы, где куратор и староста смогут отмечать присутствие и выгружать отчет.

Библиография

1. FastAPI [электронный ресурс]. Документация FastAPI. – URL: <https://fastapi.tiangolo.com> (дата обращения: 22.02.2023).
2. Python 3.11.3 documentation [электронный ресурс]. Документация Python. URL: <https://docs.python.org/3/> (дата обращения 23.01.2023)
3. SQLAlchemy 2.0 Documentation [электронный ресурс]: Документация SQLAlchemy. – URL: <https://docs.sqlalchemy.org/en/20/> (дата обращения 24.03.2023)
4. Git [электронный ресурс]. Документация Git – URL: <https://git-scm.com/doc> (дата обращения 18.05.2023)
5. Figma [электронный ресурс]. Прототипирование проектов. – URL: <https://www.figma.com/> (дата обращения 12.12.2022)

Приложение А

Техническое задание

Министерство образования Новосибирской области

ГБПОУ НСО «Новосибирский авиационный технический колледж

имени Б.С. Галущака»

Разработка мобильного личного кабинета студента НАТК

НАТКиГ.200600.010.000 ПЗ

Выполнил:

Молотков Д.К.

2023

**Содержание**

[Введение 39](#_Toc121149826)

[1 Назначение разработки 40](#_Toc121149827)

[2 Требования к мобильному приложению 41](#_Toc121149828)

[2.1 Требования к функциональным характеристикам 41](#_Toc121149829)

[2.2 Требования к надёжности 41](#_Toc121149830)

[2.3 Условия эксплуатации 42](#_Toc121149831)

[2.4 Требования к составу и параметрам технических средств 42](#_Toc121149832)

[2.5 Требования к информационной и программной совместимости 42](#_Toc121149833)

[2.6 Требования к защите информации 42](#_Toc121149834)

[2.7 Требования к маркировке и упаковке 42](#_Toc121149835)

[3 Требования к программной документации 43](#_Toc121149836)

[4 Технико-экономические показатели 44](#_Toc121149837)

[5 Стадии и этапы разработки 45](#_Toc121149838)

[6 Порядок контроля и приёмки 46](#_Toc121149839)

Введение

Настоящее техническое задание распространяется на разработку информационного программного продукта «Разработка мобильного личного кабинета студента НАТК», используемого для получения справочной информации студента учебного заведения.

Наименование приложения: «Мой НАТК».

Краткая характеристика области применения: приложение для студентов учебного заведения.

Условные обозначения и сокращения:

БД – База данных;

ИС – Информационная система;

ОС – Операционная система;

Основанием для проведения разработки является Протокол № Уч-34/1  
 от «7» марта 2023 года.

Наименование темы разработки – «Разработка мобильного приложения «Личный кабинет студента НАТК»».

Условное обозначение темы разработки – «Мой НАТК».

1 Назначение разработки

Основное назначение мобильного приложения заключается в:

* Получении информации об учебном процессе.

Лица, которые могут работать с данной Системой:

* пользователь – просмотр информации.

2 Требования к мобильному приложению

2.1 Требования к функциональным характеристикам

Требования к составу выполняемых функций:

* авторизация пользователей;
* отображение расписания учебных занятий учебного заведения;

Входные данные должны быть организованы в виде вводимого, в специальную форму, текста (логин, пароль), соответствующего определённому шаблону. Данные, вводимые вручную, проверяются на корректность.

2.2 Требования к надёжности

Обеспечение устойчивого функционирования должно выполняться при соблюдении некоторых условий:

* организация бесперебойного питания оборудования пользователя;
* использование лицензионного программного обеспечения;
* организация стабильного интернет-соединения.

Приложение должно контролировать входную информацию:

* соблюдение типов данных при заполнении полей.

Время восстановления после отказа, вызванного сбоем электропитания технических средств, не фатальным сбоем ОС или файловой системы, не должно превышать 5 минут при соблюдении условий эксплуатации технических и программных средств и правильной настройки операционной системы.

2.3 Условия эксплуатации

Обслуживание ИС включает в себя:

1. информационное обслуживание – ввод и редактирование информации БД;
2. системное администрирование БД ИС.

2.4 Требования к составу и параметрам технических средств

Для работы приложения на устройстве требуется: объем оперативной памяти не менее 256 МБ, стабильное подключение к сети, версия Android на устройстве 7 или выше.

2.5 Требования к информационной и программной совместимости

Проектирование взаимодействия с файловой системой должно быть выполнено в рамках разработки курсового проекта. При разработке взаимодействия с файловой системой должен быть использован язык программирования Python.

2.6 Требования к защите информации

Доступ к информации БД предоставляется только администратору базы данных.

2.7 Требования к маркировке и упаковке

Логотип приложения НАТК.

3 Требования к программной документации

Состав программной документации должен включать в себя:

* техническое задание;
* пояснительная записка.

4 Технико-экономические показатели

Экономические преимущества разработки и ориентировочная экономическая эффективность не рассчитывается.

5 Стадии и этапы разработки

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название этапа | Срок, даты | Отчётность |
| 1 | Исследование предметной области | 08.04.2023-9.04.2023 | Пояснительная записка |
| 2 | Выбор моделей, описывающих предметную область | 10.04.2023-14.04.2023 | Пояснительная записка |
| 3 | Разработка технического задания | 15.04.2023-16.04.2023 | Техническое задание |
| 4 | Изучение Python, Xamarin.Forms и SQLAlchemy | 17.04.2023-  20.04.2023 | Пояснительная записка |
| 5 | Анализ требований и уточнение спецификаций | 21.04.2023 | Спецификации программного обеспечения |
| 6 | Проектирование структуры мобильного приложения, проектирование компонентов (технический проект) | 22.04.2023-30.04.2023 | Схема структурная системы и спецификации компонентов |
| 7 | Кодирование клиентской части | 01.05.2023-12.05.2023 | Программный  Продукт |
| 8 | Тестирование компонентов.  Сборка и комплексное тестирование | 13.05.2023 | Тексты программных компонентов |
| 9 | Разработка программной  документации | 14.05.2023 – 22.05.2023 | Программная  документация |
| 10 | Защита | 29.05.2023 |  |

Таблица 1 – Стадии разработки

6 Порядок контроля и приёмки

Виды испытаний – защита курсового проекта.

Общее требования к приёмке:

* техническое задание;
* пояснительная записка;
* программный продукт;
* презентация.