МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РЕСПУБЛИКИ КАЗАХСТАН

Казахский национальный исследовательский технический университет
имени К.И. Сатпаева

Институт автоматики и информационных технологий

Кафедра «Программная инженерия»

Қыстақов Ерсұлтан Нұржігітұлы

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к дипломному проекту

Образовательная программа 6B06102 – Computer Science

Алматы 2025

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РЕСПУБЛИКИ КАЗАХСТАН

Казахский национальный исследовательский технический университет имени К.И.Сатпаева

Институт автоматики и информационных технологий

Кафедра «Программная инженерия»

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой ПИ
канд. техн. наук, ассоц.профессор
_____ Ф.Н. Абдолдина
“_05_” _06_ 2025 г.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту

На тему: «Приложение «Студенческий Справочник» будет позволять студентам легко запрашивать и получать различные справки, необходимые для получения пособий, военного учета и других административных целей»

Образовательная программа: 6B06102 Computer Science

Выполнил — Қыстаков Ерсұлтан Нұржігітұлы

Рецензент
PhD, ассоциированный профессор
_____ Г.Б. Кашаганова
"_05_"_____06_____2025 г.

Научный руководитель
старший преподаватель, магистр
_____ А.М. Байгаринов
"_05_"_____06_____2025 г.

Алматы 2025

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РЕСПУБЛИКИ КАЗАХСТАН

Казахский национальный исследовательский технический университет имени К.И.Сатпаева

Институт автоматики и информационных технологий

Кафедра «Программная инженерия»

УТВЕРЖДАЮ
Заведующий кафедрой ПИ
канд. тех. наук, ассоц.профессор
_____ Ф.Н. Абдолдина
«03» 06 2025 г.

ЗАДАНИЕ
на выполнение дипломного проекта

Обучающемуся: Қыстақов Ерсұлтан Нұржігітұлы
Тема: Приложение «Студенческий Справочник» будет позволять студентам легко запрашивать и получать различные справки, необходимые для получения пособий, военного учета и других административных целей
Утверждена приказом проректора по академической работе: 1804-до

№ 26-П/Ө	*от «29» января 2025 г.*
Срок сдачи законченного проекта	*«30» мая 2025 г.*

Исходные данные к дипломному проекту:
А) Анализ предметной области, анализ аналогичных проектов;
Б) Разработка технического задания;
В) Разработка архитектуры ПО;
Г) Проектирование и разработка понятного и интерактивного дизайна;
Д) Разработка и тестирование ПО;
Перечень подлежащих разработке в дипломном проекте вопросов: (с точным указанием обязательных чертежей): *представлены 7-8 слайда презентации.*
Рекомендуемая основная литература: *из 27 наименований.*

ГРАФИК

подготовки дипломного проекта

Наименование разделов, перечень разрабатываемых вопросов	**Сроки представления научному руководителю и консультантам**	**Примечание**
1. Анализ предметной области, разработка технического задания	08.04.2025	Выполнено
2. Выбор технологий для разработки	13.04.2025	Выполнено
3. Разработка логики взаимодействия	20.04.2025	Выполнено
4. Разработка функционала системы	11.04.2025	Выполнено
5. Тестирование и оптимизация	18.05.2025	Выполнено
6. Написание пояснительной записки к дипломному проекту	25.05.2025	Выполнено

Подписи

консультантов и нормоконтролера на законченную дипломный проект
с указанием относящихся к ним разделов проекта

Наименования разделов	**Консультанты, И.О.Ф. (уч. степень, звание)**	**Дата подписания**	**Подпись**	**Оценка**
Программное обеспечение	Шаханов А.Р. ст. преподаватель, магистр.	3.06.25		90
Нормоконтролер	Имиматдинова К.Ф. ст. преподаватель, магистр.	30.05.2025		91

Научный руководитель Байгаринов А.М.

Задание принял к исполнению обучающийся Кыстаков Е.Н.

Дата «30» мая 2025г.

АҢДАТПА

Бұл дипломдық жобада студенттерге әртүрлі мақсаттарға қажетті анықтамаларды (әлеуметтік жәрдемақылар, әскери есепке тұру, т.б.) электронды түрде оңай әрі жылдам алуға мүмкіндік беретін мобильді қосымша жасалды. Қосымша университеттің қазіргі ақпараттық жүйесімен интеграциялану арқылы деректер базасына тікелей қол жеткізуді қамтамасыз етеді. Жоба барысында қосымшаның архитектурасы, пайдаланушы интерфейсі, функционалдық модульдері және интеграция әдістері әзірленді. Сондай-ақ, бағдарламалық қамсыздандыру тестіленіп, оның сенімділігі мен қолдануға ыңғайлылығы тексерілді.

АННОТАЦИЯ

В данной выпускной квалификационной работе разработано мобильное приложение, позволяющее студентам легко и быстро получать необходимые справки (для получения социальных пособий, постановки на воинский учёт и других административных целей). Приложение интегрируется с существующей информационной системой университета, что позволяет напрямую взаимодействовать с базой данных. В рамках проекта была спроектирована архитектура приложения, реализован пользовательский интерфейс, проработаны механизмы интеграции и проведено функциональное тестирование. Приложение упрощает административные процессы и повышает удобство для студентов.

ABSTRACT

This graduation project presents the development of a mobile application designed to help students quickly and easily request and receive various official certificates (such as those required for social benefits, military registration, and other administrative purposes). The application is integrated with the university's existing information system, allowing secure and direct access to the database. The project includes the design of the application's architecture, development of user interface and integration mechanisms, as well as functional testing. The solution simplifies administrative procedures and improves convenience for students.

СОДЕРЖАНИЕ

	Введение	9
1	Анализ предметной области и постановка задачи	10
1.1	Актуальность автоматизации административных процессов в вузах	10
1.2	Обзор существующих решений	10
1.3	Постановка задачи	11
1.3.1	Техническое задание	11
1.3.2	Функциональные требования	12
1.3.3	Нефункциональные требования	13
2	Проектирование информационной системы	15
2.1	Проектирование архитектуры приложения	15
2.2	Структура БД	16
2.3	Разработка дизайна	18
2.4	Обоснование выбора технологий и инструментов	20
3	Разработка	21
3.1	Процесс разработки	21
3.2	Тестирование	22
3.3	Развертывание	25
	Заключение	26
	Список использованной литературы	27
	Приложение А. Техническое задание	28
	Приложение Б. Листинг программы	30

ВВЕДЕНИЕ

В современном мире цифровизация охватывает все сферы жизни, включая образование. Особенно актуальным становится создание сервисов, которые упрощают взаимодействие студентов с учебными заведениями. Одной из типичных административных задач, с которой регулярно сталкиваются студенты, является получение различных справок — для военкомата, социальных пособий, подтверждения статуса учащегося и т.д. Этот процесс, как правило, требует личного присутствия, бумажной волокиты и временных затрат как со стороны студентов, так и сотрудников вуза.

Целью настоящего дипломного проекта является разработка мобильного приложения, которое автоматизирует процесс получения справок и делает его доступным в несколько кликов. Приложение предоставляет пользователю возможность подать заявку, выбрать нужный тип справки и получить её в электронном виде. Особое внимание уделено интеграции приложения с существующей системой университета, что позволяет использовать уже имеющуюся базу данных и поддерживать актуальность информации.

Разработка такого приложения способствует улучшению цифровой инфраструктуры учебного заведения, повышает удобство для студентов и снижает нагрузку на административный персонал. Кроме того, гибкая архитектура решения позволяет масштабировать его и использовать в других образовательных учреждениях.

1 Анализ предметной области и постановка задачи

1.1 Актуальность автоматизации административных процессов в вузах

В условиях стремительного развития цифровых технологий образовательные учреждения сталкиваются с необходимостью модернизации внутренних процессов. Особенно важной задачей становится оптимизация административных процедур, таких как выдача справок студентам. Традиционный способ получения справки предполагает личное обращение в учебный отдел, заполнение бумажных заявлений и ожидание обработки запроса, что нередко сопровождается потерей времени и неудобствами как для студента, так и для сотрудников вуза.

Современные студенты, будучи активными пользователями цифровых сервисов, ожидают от учебных заведений аналогичного уровня комфорта и доступности. Поэтому необходимость создания мобильных решений, которые позволяют автоматизировать процесс получения справок, становится особенно актуальной. Такие приложения значительно ускоряют обслуживание, снижают нагрузку на административный персонал и минимизируют человеческий фактор.

Кроме того, в условиях дистанционного и гибридного форматов обучения мобильное приложение становится не просто удобным инструментом, а необходимостью. Это решение способствует переходу вуза к модели "цифрового университета", где основное взаимодействие со студентами происходит через современные IT-сервисы.

1.2 Обзор существующих решений

В последние годы на рынке образовательных технологий появилось множество решений, ориентированных на автоматизацию документооборота в высших учебных заведениях. Среди них встречаются как крупные комплексные системы управления университетами (LMS и ERP), так и специализированные модули для обработки заявок на справки.

Примеры таких решений включают:

- Platonus, используемый в ряде казахстанских вузов, который обеспечивает базовый функционал для работы с учебными данными, но часто требует отдельного обращения в деканат для оформления справок.
- Univer, внедряемый в ряде вузов СНГ, предоставляет модуль для электронного документооборота, однако не всегда предусматривает интеграцию с мобильными платформами.

- Некоторые вузы разрабатывают собственные закрытые веб-платформы, через которые можно заказать справку, но интерфейс и доступ к таким системам часто ограничены внутренней сетью университета.

Основным недостатком большинства из перечисленных решений является отсутствие нативных мобильных приложений, интуитивного интерфейса и возможности гибкой интеграции с внешними системами. Кроме того, нередки случаи, когда справка оформляется вручную, даже если заявка была подана в электронном виде, что сводит на нет преимущества цифровизации.

Таким образом, на сегодняшний день рынок всё ещё испытывает дефицит адаптивных, лёгких в использовании и по-настоящему мобильных решений для автоматизированной генерации и выдачи справок. Это открывает возможности для разработки специализированного приложения, полностью ориентированного на потребности студентов и способного легко интегрироваться с существующей инфраструктурой вуза.

1.3 Постановка задачи

На основе анализа текущего состояния и существующих решений в области автоматизации выдачи справок, можно сформулировать ключевую задачу дипломного проекта — разработка мобильного приложения, обеспечивающего удобный и быстрый способ получения студентами различных видов справок, востребованных в административных и социальных целях.

Целью проекта является создание интегрируемого программного продукта, который сможет подключаться к уже функционирующей информационной системе вуза. Особое внимание при разработке будет уделено вопросам юзабилити, безопасности и соответствию требованиям учебного учреждения.

1.3.1 Техническое задание

Приложение должно обеспечивать ряд ключевых функциональных возможностей, направленных на упрощение и автоматизацию процесса получения справок для студентов. В первую очередь, система обязана предоставлять пользователю удобный и структурированный список всех доступных видов справок. Это необходимо для того, чтобы студент мог быстро ознакомиться с возможными вариантами и выбрать именно ту справку, которая ему необходима — будь то справка для военкомата, для получения социальной помощи и т.д.

Кроме того, особое внимание уделяется разработке интуитивно понятного и простого интерфейса, который позволит студенту без лишних усилий подать заявку на получение справки. Процесс подачи должен занимать минимальное количество времени и не требовать от пользователя специальных технических знаний, что особенно важно в условиях широкой аудитории пользователей.

После подачи заявки приложение должно формировать и отправлять соответствующий запрос на серверную часть действующей информационной системы университета. Сервер, в свою очередь, обрабатывает запрос, взаимодействует с базой данных и отвечает готовым документом.

Готовая справка формируется в формате PDF, что является наиболее универсальным и удобным способом хранения и передачи официальных документов. Полученный PDF-документ должен быть доступен для скачивания и открываться непосредственно из приложения.

Также крайне важным компонентом является реализация модуля авторизации и идентификации студента. Это необходимо для обеспечения безопасности и конфиденциальности персональных данных. Только после успешной авторизации пользователь получает доступ ко всем функциям системы, но это зависит от метода интеграции.

Таким образом, приложение должно быть не только функциональным, но и надёжным, удобным и безопасным инструментом взаимодействия студентов с университетской административной системой.

1.3.2 Функциональные требования

В рамках разработки мобильного приложения особое внимание уделяется функциональным требованиям, определяющим основной перечень возможностей системы, необходимых для её эффективной работы и удовлетворения пользовательских ожиданий.

Одним из ключевых требований является создание наглядного и интуитивно понятного интерфейса, предназначенного для удобного отображения информации и обеспечения простоты взаимодействия пользователя с системой. Интерфейс должен быть максимально адаптирован под потребности студентов, включать понятную навигацию, логичную структуру экранов и чётко оформленные элементы управления.

Среди первостепенных функций предусмотрена регистрация и авторизация пользователя через студенческий аккаунт. Данная процедура позволяет идентифицировать личность пользователя и ограничить доступ к функционалу только для действующих студентов. Это важно как с точки зрения информационной безопасности, так и для корректной обработки персональных данных.

После успешной авторизации студенту предоставляется возможность ознакомиться со списком доступных справок. Этот список формируется на

основании заранее заданных шаблонов и может включать такие справки, как подтверждение обучения, справка для предоставления в военкомат, для получения социальной поддержки и другие. Отображение перечня справок должно быть реализовано в удобном и структурированном виде, с возможностью выбора нужной категории.

Далее студент может перейти к подаче заявки на получение выбранной справки. При оформлении заявки пользователь указывает цель запроса, тип необходимой справки, а также другие требуемые параметры (например, место предоставления справки или дополнительную информацию, связанную с запросом). Система должна обрабатывать эту информацию и передавать её на серверную часть для дальнейшей генерации документа.

После завершения обработки запроса и формирования справки, пользователю предоставляется возможность просмотреть готовый документ прямо в интерфейсе приложения. Также реализуется функция загрузки документа в формате PDF на устройство студента, что позволяет использовать справку в электронном виде или при необходимости распечатать её.

Особое внимание уделяется внедрению QR-кода в каждую сгенерированную справку. QR-код содержит уникальную ссылку на веб-страницу, где размещена цифровая копия выданной справки. Это позволяет третьим лицам (например, сотрудникам государственных органов) оперативно проверить подлинность документа, отсканировав QR-код. Такая технология значительно повышает доверие к цифровому документу, обеспечивает прозрачность и исключает возможность подделки.

Таким образом, функциональные возможности приложения направлены на полную автоматизацию и цифровизацию процесса выдачи справок, повышение удобства для студентов и обеспечение надежной верификации документов.

1.3.3 Нефункциональные требования

Помимо функциональных возможностей, разработка мобильного приложения требует учёта и выполнения ряда нефункциональных требований, обеспечивающих общее качество системы, удобство её использования, безопасность и устойчивость к нагрузкам.

Одним из приоритетных аспектов является обеспечение высокой доступности и стабильности работы приложения. Система должна быть готова к постоянной эксплуатации в круглосуточном режиме с минимальными перерывами на техническое обслуживание. Это особенно важно, учитывая необходимость предоставления студентам возможности подавать заявки на справки в любое удобное время. Приложение должно корректно функционировать в различных условиях, в том числе при нестабильном интернет-соединении, обеспечивая устойчивость и корректную обработку пользовательских запросов.

Не менее важным является наличие интуитивно понятного пользовательского интерфейса. Приложение должно быть ориентировано на широкую аудиторию студентов, среди которых могут быть пользователи с различным уровнем цифровой грамотности. Поэтому все элементы интерфейса — от главного меню до форм подачи заявок — должны быть реализованы в соответствии с принципами удобства, логики и простоты взаимодействия. Интерфейс должен исключать двусмысленности и минимизировать количество действий, необходимых для достижения цели.

Отдельное внимание в проектировании приложения уделяется вопросам безопасности. В системе предусмотрены механизмы защиты персональных данных студентов, а также безопасность при передаче и хранении информации. Все данные, обрабатываемые в рамках приложения (включая персональные сведения, заявки, справки и т.д.), должны быть зашифрованы и передаваться по защищённым протоколам. Это исключает возможность несанкционированного доступа и утечек информации, что особенно важно в условиях работы с конфиденциальной информацией.

Кроме того, приложение должно обладать способностью к масштабированию. Это означает, что архитектура и программная реализация должны быть спроектированы таким образом, чтобы при увеличении числа пользователей, объёма данных или новых функциональных модулей система могла быть адаптирована без потери производительности. Возможность масштабирования обеспечит долгосрочную жизнеспособность проекта и позволит расширять его возможности в будущем, например, при интеграции с другими университетскими сервисами или государственными платформами.

Таким образом, реализация нефункциональных требований играет важную роль в обеспечении общего качества программного продукта и его соответствия современным требованиям к мобильным информационным системам.

2 Проектирование информационной системы

2.1 Проектирование архитектуры приложения

Архитектура разрабатываемого мобильного приложения основывается на классической клиент-серверной модели, что позволяет достичь гибкости, расширяемости и лёгкости в сопровождении программного продукта. Данная модель обеспечивает чёткое разделение обязанностей между клиентской частью, установленной на мобильном устройстве пользователя, и серверной частью, расположенной в инфраструктуре университета. Такой подход позволяет эффективно распределять вычислительные ресурсы и обеспечить надежную работу приложения при взаимодействии с внутренними информационными системами вуза.

Клиентом в данной архитектуре выступает мобильное приложение, установленное на устройстве студента. Оно отвечает за отображение пользовательского интерфейса, обработку взаимодействий с пользователем, формирование запросов и получение справок. Через приложение студенты могут авторизоваться, просматривать доступные виды справок, а также получать готовые документы в формате PDF. При этом клиент не обрабатывает или хранит чувствительные данные — вся логика, связанная с управлением данными, вынесена на сервер.

Серверной частью является уже существующая информационная система университета, на базе которой работает внутренний API. Именно через этот интерфейс происходит передача запросов от мобильного клиента к серверу и получение ответов в структурированном виде. Такой подход позволяет использовать существующие базы данных и модули, не создавая дополнительных нагрузок на текущую инфраструктуру.

Для безопасного и корректного взаимодействия между мобильным приложением и сервером предусмотрен слой интеграции. Этот слой включает в себя методы, протоколы и механизмы аутентификации, которые обеспечивают надёжную передачу данных между компонентами системы. В качестве основного протокола взаимодействия используется REST API, обеспечивающий простоту в реализации, масштабируемость и поддержку большинства современных мобильных платформ.

При проектировании архитектуры предусмотрено два варианта интеграции мобильного приложения с университетской системой. Первый вариант предполагает интеграцию через API, при которой приложение функционирует как внешнее решение и взаимодействует с внутренними сервисами университета через безопасный интерфейс. Такой подход является менее инвазивным и позволяет развивать мобильное приложение независимо от основной системы.

Второй вариант предполагает полную интеграцию путём внедрения исходного кода мобильного клиента в существующее веб-приложение

университета. В этом случае мобильное решение становится неотъемлемой частью общей экосистемы вуза, что может упростить поддержку и обеспечить более тесную связь с другими модулями университетской платформы.

Таким образом, архитектура приложения направлена на обеспечение надёжного, безопасного и эффективного взаимодействия между пользователем и университетскими информационными системами, с возможностью гибкой интеграции в зависимости от технических и организационных требований.

2.2 Структура БД

В рамках реализации дипломного проекта создание отдельной базы данных не требуется, поскольку приложение разрабатывается как дополнение к уже функционирующей информационной системе университета. Таким образом, мобильное приложение будет использовать существующую централизованную базу данных вуза, в которой уже содержится актуальная информация о студентах, заявках и выданных справках.

Несмотря на отсутствие необходимости в создании новой базы данных, при проектировании и разработке необходимо учитывать особенности структуры и логики хранения данных в текущей системе. Одной из ключевых таблиц, с которой осуществляется взаимодействие, является таблица Students, содержащая персональные данные обучающихся. В данной таблице, как правило, представлены следующие поля: ФИО (фамилия, имя, отчество студента), уникальный идентификатор (ID), номер курса, учебная группа и другие атрибуты, необходимые для генерации справок.

Работа мобильного приложения строится на основе клиент-серверного взаимодействия, при котором к базе данных напрямую подключается только серверная часть. Клиентское мобильное приложение не получает прямого доступа к базе данных, что обеспечивает необходимый уровень безопасности и целостности данных. Вместо этого клиент отправляет структурированные запросы через API, а сервер возвращает запрошенные данные в безопасной форме.

На стороне клиента реализовано только временное хранение данных, преимущественно в виде кэширования. Это необходимо для повышения отзывчивости интерфейса и уменьшения количества сетевых запросов, особенно в условиях нестабильного интернет-соединения. Такие данные могут включать список типов справок, информацию о ранее отправленных заявках и другие вспомогательные сведения, не требующие постоянного соединения с сервером.

Для целей тестирования и отладки, а также при разработке функционала в изолированной среде, была создана модель, имитирующая взаимодействие с базой данных университета. Для этого использовалась технология Entity Framework Core — современный ORM-инструмент, предоставляющий удобный и гибкий способ работы с базами данных в .NET-среде. С его помощью были

разработаны модели данных, отражающие структуру таблиц, а также реализованы методы для выполнения CRUD-операций (создание, чтение, обновление, удаление).

Таким образом, структура базы данных, используемая в проекте, строится на существующей инфраструктуре вуза, что позволяет обеспечить высокую степень интеграции, минимизировать избыточность данных и сохранить единый источник достоверной информации.

```
using Microsoft.EntityFrameworkCore;
using References_Int_.Models;

namespace References_Int_.Data
{
    Ссылок: 12
    public class UniversityContext : DbContext
    {
        Ссылок: 0
        public UniversityContext(DbContextOptions<UniversityContext> options)
            : base(options) { }

        Ссылок: 8
        public DbSet<Student> Students { get; set; }

        Ссылок: 0
        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);

            modelBuilder.Entity<Student>().HasData(

                new Student
                {
                    Id = 1,
                    FullName = "Қыстақов Ерсұлтан Нұржігітұлы",
                    Faculty = "Computer Science",
                    Course = 3,
                    Group = "6B06102",
                    Education = "очная",
                    Training_Period = "Период обучения с 01.09.2022 года по 30.06.2025 года ",
                    Birth = "09.12.2002 г.р.",
                },

                new Student
                {
                    Id = 2,
                    FullName = "Иван Иванов Иванович",
                    Faculty = "Information Security",
                    Course = 1,
                    Group = "999999",
                    Education = "заочная",
                    Training_Period = "Период обучения с 01.09.2024 года по 30.06.2027 года ",
                    Birth = "30.11.2004 г.р.",
                }
```

Рисунок 1 – Структура базы данных

2.3 Разработка дизайна

Разработка пользовательского интерфейса (UI) является важным этапом проекта, поскольку от удобства и эстетичности дизайна напрямую зависит комфорт пользователей при взаимодействии с мобильным приложением. Интерфейс должен быть интуитивно понятным, легким для восприятия и отвечать современным требованиям мобильной разработки.

Дизайн приложения реализуется в минималистичном стиле, что позволяет сконцентрировать внимание пользователя исключительно на функционале без лишних отвлекающих элементов. Такой подход обеспечивает быструю навигацию по разделам приложения, простоту восприятия текста и иконок, а также легкость освоения даже для пользователей без технического опыта.

Для обеспечения универсальности и кроссплатформенности дизайн адаптируется под различные разрешения экранов и ориентации устройств. В процессе разработки используются принципы адаптивного дизайна, позволяющие корректно отображать интерфейс как на современных смартфонах с высоким разрешением, так и на устройствах с меньшими экранами.

Особое внимание уделяется применению современных дизайнерских подходов, в частности Material Design, разработанного компанией Google. Этот подход включает использование стандартных визуальных компонентов, таких как карточки, кнопки, панели навигации и поля ввода, что способствует унификации внешнего вида и улучшает взаимодействие с интерфейсом. Использование теней, анимаций, плавных переходов и согласованной цветовой палитры делает интерфейс не только функциональным, но и визуально приятным.

Дополнительно продумываются все элементы пользовательского пути — от авторизации до получения справки. Все кнопки, формы и уведомления проектируются таким образом, чтобы минимизировать количество действий со стороны пользователя и обеспечить логичную последовательность шагов. Также внедряются элементы визуальной обратной связи — индикаторы загрузки, сообщения об успешных действиях и ошибках.

Таким образом, дизайн мобильного приложения ориентирован на максимальную доступность и удобство, соответствие стандартам UX/UI и высокое качество визуального восприятия, что в совокупности повышает пользовательскую лояльность и эффективность взаимодействия с системой.

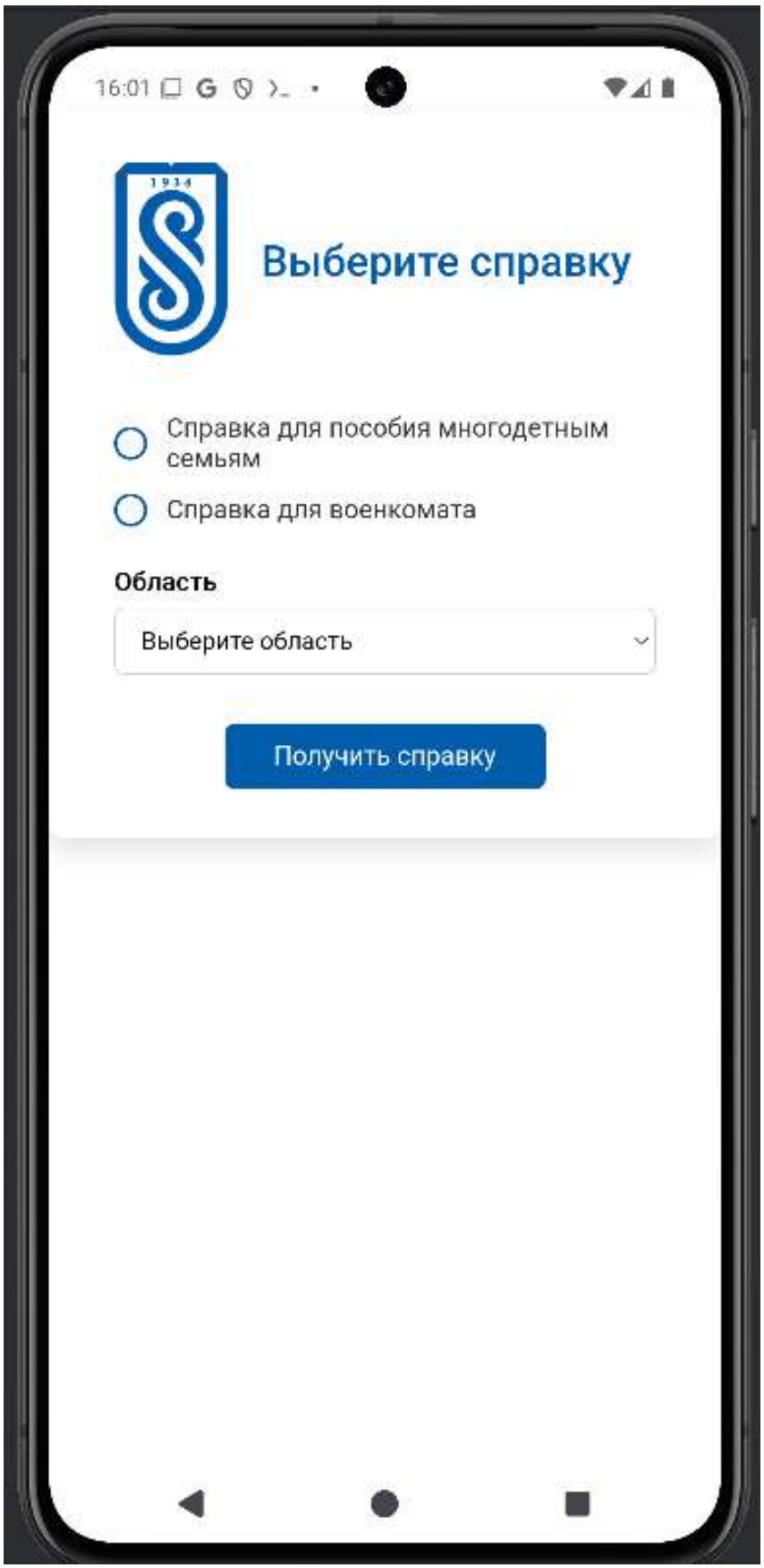

Рисунок 2 – Дизайн приложения

2.4 Обоснование выбора технологий и инструментов

Одним из ключевых факторов успешной реализации проекта является выбор технологий и инструментов, которые обеспечат надёжность, производительность, безопасность и совместимость с действующей информационной системой вуза. Поскольку университетская система уже функционирует на основе стека технологий .NET (C#) и Angular, в рамках дипломного проекта было принято решение придерживаться аналогичного стека. Это решение позволяет не только обеспечить легкую интеграцию, но и избежать дублирования логики, повысить повторное использование компонентов и сократить время на внедрение.

Для реализации клиентской части проекта был выбран Angular — современный фреймворк на языке TypeScript, обеспечивающий высокую производительность, модульность и расширяемость интерфейса. Angular отлично подходит для построения одностраничных приложений и обладает широким сообществом и поддержкой. Дополнительно применён Angular Material — библиотека UI-компонентов, соответствующая принципам Material Design. Это позволило адаптировать интерфейс под мобильные устройства и создать визуально привлекательное и удобное приложение.

Серверная часть разрабатывается с использованием языка программирования C# и платформы .NET Core, что обеспечивает высокую производительность и надёжность серверных процессов. Для реализации взаимодействия между клиентом и сервером используется архитектура REST API, где запросы передаются по защищённому протоколу HTTPS, а структура данных формируется в формате JSON, что является стандартом для обмена информацией в веб-приложениях.

Генерация справок осуществляется в формате PDF, подходящий для хранения, распространения и проверки подлинности документов. Также рассматривается возможность интеграции QR-кода в PDF-документ для проверки оригинальности справки путём перехода по уникальной ссылке.

Для тестирования и документирования API-интерфейсов применяются инструменты Swagger и Postman. Swagger позволяет автоматически генерировать документацию по всем REST-запросам и визуализировать методы работы API. Postman, в свою очередь, активно используется в процессе ручного тестирования запросов, проверки авторизации и обработки ответов от сервера.

Таким образом, выбранный технологический стек обеспечивает устойчивую и масштабируемую архитектуру проекта, позволяет поддерживать современные требования к безопасности и удобству работы с системой, а также предоставляет возможность гибкой доработки и расширения функциональности в будущем.

3 Разработка

3.1 Процесс разработки

Процесс разработки приложения был организован поэтапно, с чётким разграничением задач и последовательной реализацией функциональных компонентов. Такой подход позволил обеспечить стабильность разработки, гибкость в доработках, а также последовательное тестирование каждого этапа.

На первом этапе была выполнена настройка архитектуры проекта. Для клиентской части был создан Angular-проект с модульной структурой, что обеспечило удобное разделение компонентов по назначению и позволило легко масштабировать интерфейс. Для серверной части был разработан Web API на платформе .NET Core с использованием языка C#, что обеспечило производительность и совместимость с действующей системой университета. Были настроены маршруты, контроллеры и сервисы для обработки запросов от клиента, работы с данными студентов и генерации справок.

Следующим важным этапом стала реализация пользовательского интерфейса. С применением Angular и библиотеки Angular Material был создан современный адаптивный интерфейс, доступный как на мобильных устройствах, так и на экранах с различным разрешением. Интерфейс был спроектирован с акцентом на простоту и удобство, чтобы минимизировать количество действий, необходимых для получения справки. Были реализованы экраны авторизации, выбора вида справки, подачи заявки и отображения статуса заявки.

Особое внимание было уделено работе с документацией. Была реализована возможность генерации документов в формате PDF на серверной стороне. При этом учитывались такие детали, как подстановка персональных данных студента, выбор нужного шаблона справки, отображение даты, печати, водяного знака, а также генерация уникального QR-кода для проверки подлинности справки. Каждому PDF-документу присваивалась уникальная ссылка, которая может быть расшифрована с помощью QR-кода и позволяет проверить действительность справки онлайн.

Отдельным этапом стала реализация системы авторизации и обеспечения безопасности. Вход в приложение осуществляется с использованием данных студенческого аккаунта. Все передаваемые данные защищаются с помощью протокола HTTPS. Также реализована проверка прав доступа для предотвращения несанкционированного получения или изменения информации. Это обеспечивает соответствие требованиям конфиденциальности и защиты персональных данных.

Таким образом, весь процесс разработки представлял собой логически структурированную последовательность этапов, каждый из которых завершался тестированием и валидацией реализованного функционала. Это обеспечило надёжную и удобную в использовании систему, интегрированную с инфраструктурой вуза.

3.2 Тестирование

В процессе разработки особое внимание уделялось тестированию функциональности и устойчивости приложения. Тестирование проводилось как вручную, так и с использованием инструментов автоматизации, что позволило выявить и устранить ошибки на ранних этапах и обеспечить высокое качество конечного продукта.

Вручную были проверены все основные пользовательские сценарии, а также поведение приложения в нестандартных и стрессовых ситуациях. Автоматизированное тестирование осуществлялось с помощью встроенных средств Angular и .NET, а также таких инструментов, как Postman и Swagger для тестирования API-запросов.

В ходе тестирования были охвачены следующие ключевые модули:

Отправка заявки на получение справки. Проверялась корректность заполнения формы, валидация данных, передача информации на сервер и получение ответа о статусе заявки. Были протестированы различные сценарии, включая подачу заявки с неполными или некорректными данными.

Получение и отображение готовой справки. Тестировалась генерация PDF-документов, их загрузка на устройство, корректное отображение содержимого, а также наличие всех обязательных элементов — ФИО, даты, QR-кода, печати.

Обработка сетевых ошибок. Особое внимание уделялось проверке поведения приложения при отсутствии соединения с интернетом или нестабильной связи. Реализованы механизмы отображения сообщений об ошибке и повторной отправки запросов.

Работа на различных устройствах и разрешениях экрана. Интерфейс приложения тестировался на устройствах с разным размером и разрешением экрана (смартфоны, планшеты). Проверялась адаптивность верстки, доступность элементов управления и корректность отображения интерфейса.

Проверка QR-кода. Отдельно тестировался механизм генерации и распознавания QR-кода, который позволяет проверить подлинность справки. Была обеспечена корректная работа сканирования кода с различных устройств и переход по ссылке для проверки справки онлайн.

Результаты тестирования показали, что приложение работает стабильно, корректно обрабатывает пользовательские действия и ошибки, а также обеспечивает необходимый уровень пользовательского опыта и безопасности.

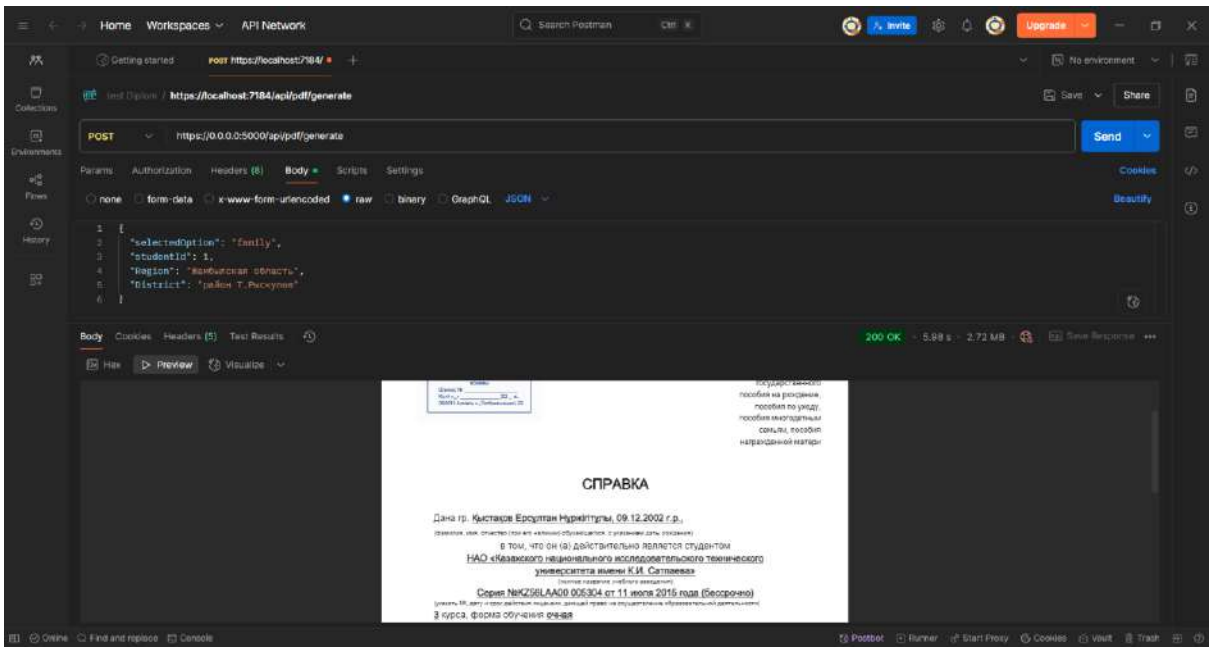

Рисунок 3 – Тестирование через Postman

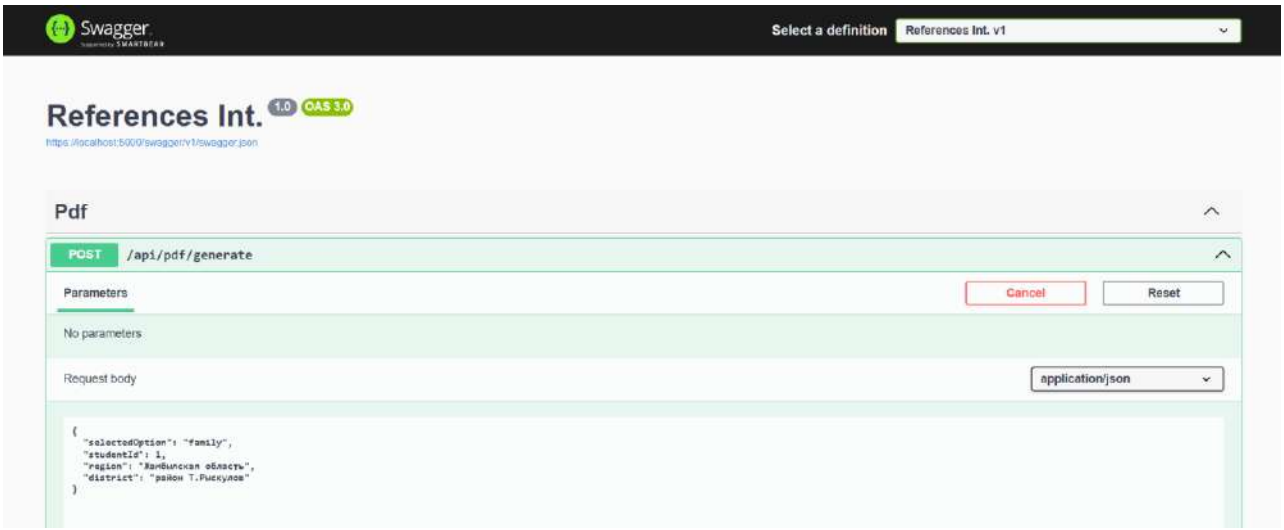

Рисунок 4.1 – Тестирование через Swagger

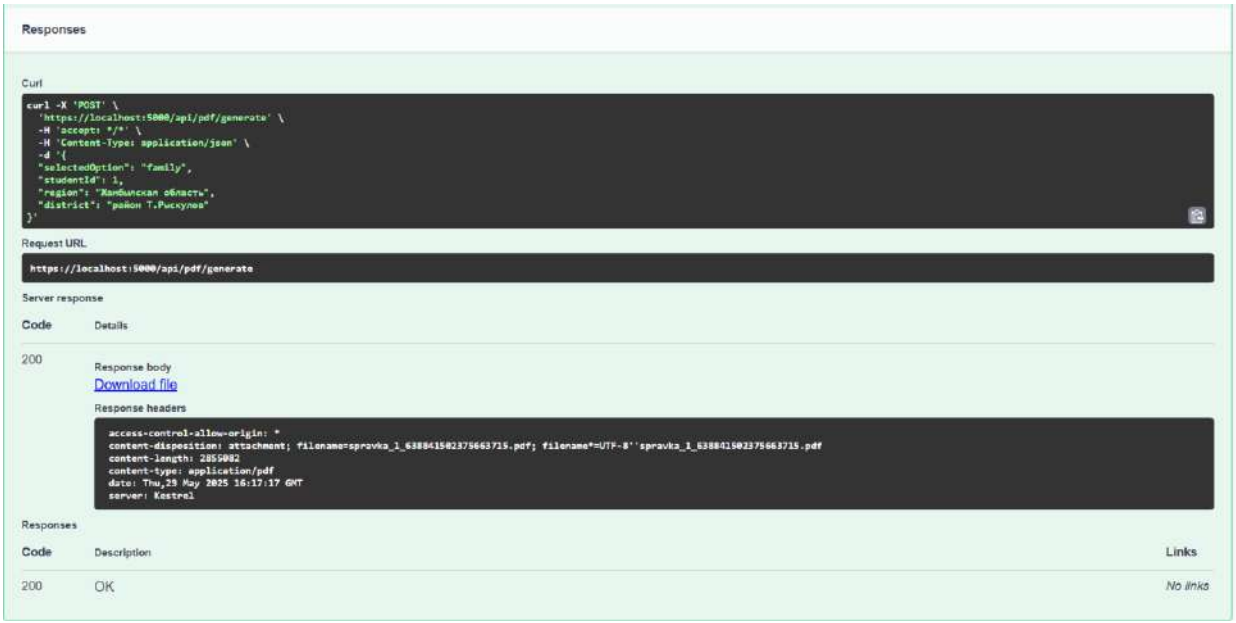

Рисунок 4.2 – Тестирование через Swagger

Қазақстан Республикасы
Білім және ғылым министрлігі
«К.И. Сәтбаев атындағы Қазақ ұлттық техникалық зерттеу университеті коммерциялық емес акционерлік қоғамы
Шығыс № _____
Күні «_» _____20__ ж.
050013 Алматы қ., Сәтбаев көшесі, 22

Приложение 29
к Правилам назначение и осуществления выплаты государственного пособия на рождение, пособия по уходу, пособия многодетным семьям, пособия награжденной матери

СПРАВКА

Дана гр. **Кыстаков Ерсултан Нуржігітұлы, 09.12.2002 г.р.,**
(фамилия, имя, отчество (при его наличии) обучающегося, с указанием даты рождения)
в том, что он (а) действительно является студентом
НАО «Казахского национального исследовательского технического университета имени К.И. Сатпаева»
(полное название учебного заведения)
Серия №KZ56LAA00 005304 от 11 июля 2015 года (бессрочно)
(указать №, дату и срок действия лицензии, дающей право на осуществление образовательной деятельности)
3 курса, форма обучения **очная**
Справка действительна на **2024/2025** учебный год.
Справка выдана для предъявления
В **Жамбылская область, район Т.Рыскулов** отделение Государственной корпорации.
Срок обучения в учебном заведении 3 года.
Период обучения с 01.09.2022 года по 30.06.2025 года
Примечание. Справка действительна на 1 год.
В случае отчисления обучающегося из учебного заведения или перевода на заочную форму обучения, руководитель учебного заведения извещает отделение Государственной корпорации по месту жительству получателя пособия.

Проректор по административной, социальной и воспитательной работе **С. Шалабаев**

Рисунок 5 – Созданный документ

3.3 Развертывание

После завершения этапов разработки и тестирования приложение было развернуто на тестовом стенде, подключенном к демонстрационной версии базы данных. Это дало возможность полноценно проверить работу всех модулей в условиях, приближенных к реальной эксплуатации, а также провести презентацию основных функций — от авторизации студента до получения готовой справки в формате PDF.

На тестовом стенде были смоделированы различные сценарии использования, в том числе одновременное обращение нескольких пользователей, имитация нестабильного интернет-соединения и взаимодействие с различными справками. Также проводилось пробное сканирование QR-кодов, вшитых в PDF-документы, с переходом на проверочную веб-страницу.

Инфраструктура развертывания включала:

- Сервер с развернутым API-приложением на .NET Core;
- Клиентскую часть на Angular, доступную через веб-браузер или мобильное устройство;
- Базу данных, подключённую к существующей системе вуза (в демонстрационном режиме);
- Средства логирования, отладки и контроля ошибок (например, встроенные логи, Postman, Swagger).

Для полноценного запуска в продуктивной среде рассматриваются два варианта развертывания:

На локальных серверах вуза. Такой подход обеспечивает полный контроль над данными, соответствие требованиям внутренней политики безопасности и отсутствие зависимости от внешних сервисов. Однако он потребует дополнительной настройки серверного оборудования, обеспечения резервного копирования и системной поддержки.

В облачной инфраструктуре. Этот вариант более гибкий и масштабируемый, позволяет быстро внедрить решение без значительных затрат на локальное оборудование. Облачные провайдеры (например, Azure или AWS) предоставляют инструменты для безопасного хранения данных, автоматического масштабирования и управления доступом.

Окончательный выбор места развертывания будет зависеть от политики безопасности университета, бюджета проекта и технических возможностей IT-отдела. При этом архитектура приложения спроектирована таким образом, чтобы обеспечить лёгкую адаптацию под любую из выбранных инфраструктур.

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломного проекта было разработано мобильное-приложение «Студенческий Справочник», предназначенное для упрощения процесса получения справок студентами. Благодаря продуманной архитектуре, приложение легко интегрируется в существующую информационную систему вуза, что позволяет не создавать отдельную инфраструктуру и использовать уже имеющуюся базу данных.

Одним из главных преимуществ решения стало его удобство и простота для конечного пользователя: студент может получить справку в несколько кликов, без необходимости личного посещения деканата или других административных служб. Это экономит время как студентов, так и сотрудников университета.

Проект также продемонстрировал возможность масштабирования и адаптации под другие вузы или организации, что делает его перспективным для дальнейшего развития. В будущем можно расширить функциональность, добавить цифровую подпись, интеграцию с госуслугами и мобильную версию приложения.

В результате, поставленные цели были достигнуты, задачи выполнены, а разработанное приложение полностью соответствует требованиям современного цифрового образования

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

Книги

1. Макконнелл С. Совершенный код. Мастер-класс — 2-е изд. — СПб.: Питер, 2019. — 896 с.
2. Фаулер М. Архитектура корпоративных приложений. — М.: Вильямс, 2021. — 560 с.
3. Freeman Э., Robson Э. Изучаем программирование на C# — СПб.: Питер, 2022. — 720 с.
4. Гудман Д., Novitski M. HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств. — СПб.: Питер, 2020. — 480 с.
5. Документация по Angular (Электронный ресурс) — Режим доступа: https://angular.io/docs
6. Документация по .NET (Электронный ресурс) — Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/
7. Паттерны проектирования. Практические примеры на C# (Электронный ресурс) — https://refactoring.guru/ru/design-patterns
8. REST API Design — Best Practices (Электронный ресурс) — https://restfulapi.net/
9. OpenAPI/Swagger Specification Documentation (Электронный ресурс) — https://swagger.io/docs/specification/about/

Приложение А

Техническое задание

1. Наименование проекта

Приложение «Студенческий Справочник» будет позволять студентам легко запрашивать и получать различные справки, необходимые для получения пособий, военного учета и других административных целей

2. Основание для разработки

Необходимость оптимизации административных процессов в учебном заведении, снижение бумажной нагрузки и повышение доступности получения справок для студентов.

3. Цель разработки

Создание специальных приложений, позволяющих студентам формировать и получать информацию (для пособий, военного учета, современных требований и т. д.) с автоматическим заполнением и проверкой состояния (через QR-код).

4. Требования к функциональности

- Авторизация и аутентификация пользователя;
- Выбор типа справки;
- Автоматическое подставление персональных данных (ФИО, курс, факультет и др.);
- Отображение водяного знака и печати в документе;
- Генерация справки в формате .pdf;
- Генерация QR-кода для проверки справки;
- Интеграция с текущей системой вуза (двумя способами: API-интеграция и модульная установка кода).

5. Нефункциональные требования

- Простой и понятный интерфейс;
- Высокая скорость отклика интерфейса;
- Безопасность хранения и передачи данных.

6. Требования к надежности

- Система должна обрабатывать запросы без сбоев;
- При отсутствии подключения к интернету должно отображаться уведомление.

7. Языки и технологии разработки

- Фронтенд: Angular, Ionic;
- Бэкенд: .NET Core, C#;
- База данных: используется тестовая база на Entity Framework Core (через API);
- Формат хранения справок: PDF.

Приложение Б

Листинг (код) программы

```
using Microsoft.AspNetCore.Mvc;
using iTextSharp.text;
using iTextSharp.text.pdf;
using Microsoft.EntityFrameworkCore;
using References_Int_.Models;
using References_Int_.Data;
using Org.BouncyCastle.Tls;
using System.Data;
using QRCoder;

[ApiController]
[Route("api/pdf")]
public class PdfController : ControllerBase
{
    private readonly UniversityContext _context;

    public PdfController(UniversityContext context)
    {
        _context = context;
    }

    [HttpPost("generate")]
    public IActionResult GeneratePdf([FromBody] PdfRequest request)
    {
        // получаем данные из input - radio
        if (string.IsNullOrEmpty(request.SelectedOption))
        {
            return BadRequest("Тип справки не выбран");
        }

        // получаем данные из input - region,  district
        var FullLocation = $"{request.Region}, {request.District}";

        var student = _context.Students.FirstOrDefault(s => s.Id == request.StudentId);
        if (student == null)
        {
            return NotFound("Студент не найден");
        }
```

Приложение Б

```
    // данные QR
    string fileName = $"spravka_{request.StudentId}_{DateTime.Now.Ticks}.pdf";
    string filePath = Path.Combine(Directory.GetCurrentDirectory(), "www-pdf-
check", "docs", fileName);
    string fileUrl = $"{Request.Scheme}://{Request.Host}/docs/{fileName}";

    using (var memoryStream = new MemoryStream())
    {
      try
      {
        var document = new Document(PageSize.A4);

        var writer = PdfWriter.GetInstance(document, memoryStream);

        writer.CloseStream = false;
        document.Open();

        // загрузка шрифта с поддержкой кириллицы
        string fontPath =
Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Fonts),
"arial.ttf");
        BaseFont baseFont = BaseFont.CreateFont(fontPath,
BaseFont.IDENTITY_H, BaseFont.EMBEDDED);
        Font smallFont = new Font(baseFont, 8, Font.NORMAL);
        Font SmallFont = new Font(baseFont, 10, Font.NORMAL);
        Font titleFont = new Font(baseFont, 18, Font.BOLD);
        Font contentFont = new Font(baseFont, 12, Font.NORMAL);
        Font boldFont = new Font(baseFont, 12, Font.BOLD);
        Font italicFont = new Font(baseFont, 10, Font.ITALIC);

        //шапка документа
        string header =
          "Приложение 29\n" +
          "к Правилам назначение и\n" +
          "осуществления выплаты\n" +
          "государственного\n" +
          "пособия на рождение,\n" +
          "пособия по уходу,\n" +
          "пособия многодетным\n" +
          "семьям, пособия\n" +
          "награжденной матери";
```

Приложение Б

```
var Header = new Paragraph(header, SmallFont)
{
   Alignment = Element.ALIGN_RIGHT,
   SpacingAfter = 30f
};
document.Add(Header);

// заголовок
var title = new Paragraph("СПРАВКА", titleFont)
{
   Alignment = Element.ALIGN_CENTER,
   SpacingAfter = 25f
};
document.Add(title);

// данные студента
Phrase studentInfo = new Phrase();
studentInfo.Add(new Chunk("Дана гр. ", contentFont));

Chunk boldUnderline_studentInfo = new Chunk($"{student.FullName},
{student.Birth},", boldFont);

//Chunk boldUnderline_studentInfo = new Chunk("Қыстақов Ерсұлтан
Нұржігітұлы, 09.12.2002. г.р.,", boldFont);
boldUnderline_studentInfo.SetUnderline(0.5f, -2f);
studentInfo.Add(boldUnderline_studentInfo);

Paragraph StudentInfo = new Paragraph(studentInfo)
{
   IndentationLeft = 30f,
   SpacingAfter = 5f
};

document.Add(StudentInfo);

// инфо студента
Paragraph docInfoStudent = new Paragraph("(фамилия, имя, отчество
(при его наличии) обучающегося, с указанием даты рождения)", smallFont)
{
   IndentationLeft = 30f,
};
```

```
        document.Add(docInfoStudent);

        // инфо студента подверждение
        Paragraph docInfoStudentConfirmed = new Paragraph("в том, что он (а)
действительно является студентом", contentFont)
        {
           Alignment = Element.ALIGN_CENTER,
        };

        document.Add(docInfoStudentConfirmed);

        // данные универа
        Paragraph university = new Paragraph()
        {
           Alignment = Element.ALIGN_CENTER,
        };
        Chunk boldUnderline_university = new Chunk("НАО «Казахского
национального исследовательского технического\n университета имени К.И.
Сатпаева»", boldFont);
        boldUnderline_university.SetUnderline(0.5f, -2f);
        university.Add(boldUnderline_university);

        document.Add(university);

        // инфо универа
        Paragraph universityInfo = new Paragraph("(полное название учебного
заведения)", smallFont)
        {
           Alignment = Element.ALIGN_CENTER,
        };

        document.Add(universityInfo);

        // лицензия университета
        Paragraph universityLicense = new Paragraph()
        {
           Alignment = Element.ALIGN_CENTER,
        };
        Chunk boldUnderline_universityLicense = new Chunk("Серия
№KZ56LAA00 005304 от 11 июля 2015 года (бессрочно)", boldFont);
        boldUnderline_universityLicense.SetUnderline(0.5f, -2f);
```

```
            universityLicense.Add(boldUnderline_universityLicense);

            document.Add(universityLicense);

            // инфо лицензия университета
            Paragraph universityLicenseInfo = new Paragraph("(указать №, дату и
срок действия лицензии, дающей право на " +
                "осуществление образовательной деятельности) ", smallFont) {
IndentationLeft = 30f, };

            document.Add(universityLicenseInfo);

            // курс, форма обучение
            Phrase courceStudent = new Phrase();
            Chunk boldUnderline_courceStudent = new Chunk($"{student.Course}",
boldFont);
            boldUnderline_courceStudent.SetUnderline(0.5f, -2f);
            courceStudent.Add(boldUnderline_courceStudent);
            courceStudent.Add(new Chunk(" курса, форма обучения ", contentFont));
            Chunk boldUnderline_studyForm = new Chunk($"{student.Education}",
boldFont);
            boldUnderline_studyForm.SetUnderline(0.5f, -2f);
            courceStudent.Add(boldUnderline_studyForm);

            document.Add(new Paragraph(courceStudent) { IndentationLeft = 30f, });

            // срок справки

            Phrase validReference = new Phrase();
            validReference.Add(new Chunk("Справка действительна на ",
contentFont));

            // функция определния даты
            DateTime now = DateTime.Now;
            string certificateValid = now.Month >= 9
               ? $"{now.Year}/{now.Year + 1}"
               : $"{now.Year - 1}/{now.Year}";
            Chunk boldUnderline_validReference = new Chunk($"{certificateValid}",
 boldFont);

            boldUnderline_validReference.SetUnderline(0.5f, -2f);
            validReference.Add(boldUnderline_validReference);
```

```
            validReference.Add(new Chunk(" учебный год.", contentFont));
            Paragraph ValidReference = new Paragraph(validReference)
            {
               IndentationLeft = 30f,
            };

            document.Add(ValidReference);

            document.Add(new Paragraph("Справка выдана для предъявления",
contentFont) { IndentationLeft = 30f, } );

            // назначение справки
            Phrase publicServiceCenters = new Phrase();
            publicServiceCenters.Add(new Chunk("В ", contentFont));
            Chunk boldUnderline_publicServiceCenters = new
Chunk($"{FullLocation}", boldFont); //Жамбылская область, район Т. Рыскулова
            boldUnderline_publicServiceCenters.SetUnderline(0.5f, -2f);
            publicServiceCenters.Add(boldUnderline_publicServiceCenters);
            publicServiceCenters.Add(new Chunk(" отделение Государственной\n
корпорации.", contentFont));
            Paragraph PublicServiceCenters = new Paragraph(publicServiceCenters)
            {
               IndentationLeft = 30f,
            };

            document.Add(PublicServiceCenters);

            document.Add(new Paragraph("Срок обучения в учебном заведении 3
года.", contentFont) { IndentationLeft = 30f, });

            // период обучение
            Phrase studyDate = new Phrase();
            Chunk Underline = new Chunk($"{student.Training_Period}",
contentFont);
            Underline.SetUnderline(0.5f, -2f);
            studyDate.Add(Underline);
            document.Add(new Paragraph(studyDate) { IndentationLeft = 30f, });

            // примечание
            document.Add(new Paragraph("Примечание. Справка действительна на
1 год. \r\n" +
```

Приложение Б

```
            "В случае отчисления обучающегося из учебного заведения или
перевода на заочную форму обучения, " +
            "руководитель учебного заведения извещает отделение
Государственной корпорации по месту жительству получателя пособия.",
            italicFont) { IndentationLeft = 30f, SpacingAfter = 60f, } );

        // подпись и штамп
        document.Add(new Paragraph("Проректор по административной,\n
социальной и воспитательной работе                                    " +
            "С. Шалабаев", boldFont) { IndentationLeft = 30f, } );

        // добавление печати
        string stampPath = Path.Combine(Directory.GetCurrentDirectory(),
"images", "stamp.png");   //@"C:\Users\kysta\source\repos\References
Int\images\stamp.png";
        Image stampImage = Image.GetInstance(stampPath);
        stampImage.ScaleAbsolute(160f, 160f);
        stampImage.SetAbsolutePosition(document.PageSize.Width - 242f, 130f);
        stampImage.Rotation = 12f * (float)Math.PI / 180f;
        document.Add(stampImage);

        // добавление подписи
        string signaturePath = Path.Combine(Directory.GetCurrentDirectory(),
"images", "signature.png"); //@"C:\\Users\\kysta\\source\\repos\\References
Int\\images\\signature.png";
        Image signatureImage = Image.GetInstance(signaturePath);
        signatureImage.ScaleAbsolute(100f, 100f);
        signatureImage.SetAbsolutePosition(document.PageSize.Width - 250f,
150f);
        document.Add(signatureImage);

        // добавление штампа
        string stampPath2 = Path.Combine(Directory.GetCurrentDirectory(),
"images", "stamp2.png"); //@"C:\Users\kysta\source\repos\References
Int\images\stamp2.png";
        Image stampImage2 = Image.GetInstance(stampPath2);
        stampImage2.ScaleAbsolute(159f, 109f);
        stampImage2.SetAbsolutePosition(document.PageSize.Width - 550f, 700f);
        document.Add(stampImage2);

        // QR code
        QRCodeGenerator qrGenerator = new QRCodeGenerator();
```

Приложение Б

```
            QRCodeData qrCodeData = qrGenerator.CreateQrCode(fileUrl,
QRCodeGenerator.ECCLevel.Q);
            PngByteQRCode pngQrCode = new PngByteQRCode(qrCodeData);
            byte[] qrCodeBytes = pngQrCode.GetGraphic(20);
            iTextSharp.text.Image qrImage =
iTextSharp.text.Image.GetInstance(qrCodeBytes);
            qrImage.ScaleAbsolute(100f, 100f);
            qrImage.SetAbsolutePosition(150f, 50f);
            document.Add(qrImage);

            document.Close();

            System.IO.File.WriteAllBytes(filePath, memoryStream.ToArray());

            memoryStream.Position = 0;

            byte[] pdfBytes  = System.IO.File.ReadAllBytes(filePath);
            return File(pdfBytes, "application/pdf", fileName);
//memoryStream.ToArray(),
        }
        catch (Exception ex)
        {
            return StatusCode(500, $"Ошибка генерации PDF: {ex.Message}" );
        }
    }
  }
}

public class PdfRequest
{
  public string SelectedOption { get; set; } = "";
  public int StudentId { get; set; }
  public string Region { get; set; } = "";
  public string District { get; set; } = "";
}
```