МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение   
высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационны систем и технологий

Кафедра Информационные системы

Дисциплина Проектирование и архитектура програмных систем

**КУРСОВОЙ ПРОЕКТ**

Тема\_\_\_\_\_Проектирование и разработка модуля

ведения графиков работы для клиники\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Выполнил студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_Кувшинов Т.А.\_\_\_\_\_\_\_\_\_\_/

подпись инициалы, фамилия

Курс\_\_\_\_\_\_третий\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Группа\_\_\_ПИбд-31\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Направление/ специальность 09.03.04 Программная инженерия

Руководитель \_\_\_\_\_\_\_ст.преподаватель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

должность, ученая степень, ученое звание

\_\_\_\_\_Корунова Надежда Владимировна\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

фамилия, имя, отчество

Дата сдачи:

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г.

Дата защиты:

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ульяновск

2025 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение   
высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационны систем и технологий

Кафедра Информационные системы

Дисциплина Проектирование и архитектура програмных систем

**ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ**

студенту \_\_ПИбд-31\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_Кувшинво Т.А.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

группа фамилия, инициалы

**Тема проекта** \_\_\_\_ Проектирование и разработка модуля

ведения графиков работы для клиники\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Срок сдачи законченного проекта** «30» мая 2025 г.

**Исходные данные к проекту** \_\_\_см. Вариант\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(базовое предприятие, характер курсового проекта (работы):

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

задание кафедры, инициативная НИР, рекомендуемая литература, материалы практики)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Содержание пояснительной записки** (перечень подлежащих разработке вопросов)

Описание предметной области

Требования к проекту

Оценка архитектурного решения

Проектное решение

Реализация приложения (контрольный пример)

**Перечень графического материала** (с точным указанием обязательных чертежей)

Диаграмма IDEF0

Диаграмма вариантов использования

Диаграмма классов

Диаграмма последовательности

Диаграмма компонентов

Диаграмма развертывания

Руководитель \_\_\_\_ст.преп.\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ /\_Н.В. Корунова\_\_\_\_\_\_/

должность подпись инициалы, фамилия

« 25 »\_февраля\_\_\_\_\_\_\_\_\_\_2025 г

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **/\_\_\_**Т.А. Кувшинов**\_\_\_\_\_\_\_\_\_\_\_\_/**

подпись инициалы, фамилия

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение   
высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

**ОТЗЫВ  
руководителя на курсовой проект**

студента \_\_\_\_\_\_Кувшинова Тимура Александровича\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

фамилия, имя и отчество

Факультет\_\_\_ИСТ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_группа\_ПИбд-31\_\_\_\_\_курс\_\_3\_\_\_\_

Дисциплина\_\_Проектирование и архитектура програмных систем\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тема проекта (работы) \_\_\_\_\_ Проектирование и разработка модуля

ведения графиков работы для клиники\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Структура пояснительной записки курсового проекта соответствует заданию и теме. В ходе работы над курсовым проектом сформировано проектное решение, предоставлены обоснование и реализация данного проектного решения в виде контрольного примера. Студент выполнил курсовой проект самостоятельно.

Программный продукт по проектному решению реализован в виде трех клиентов и сервера для обращения к БД.

Работа выполнена и сдана по графику в установленный срок.

Руководитель\_\_ ст.преп.\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ /Н.В. Корунова /

должность, учёная степень, ученое звание подпись инициалы, фамилия

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г.

**Оглавление**

[Введение 5](#_Toc196249192)

[Описание предметной области 6](#_Toc196249193)

[Требования к проекту 7](#_Toc196249194)

[Оценка архитектурного решения 10](#_Toc196249195)

[Проектное решение 16](#_Toc196249196)

[Реализация приложения 21](#_Toc196249197)

[Выводы по реализации приложения на основании проекта 24](#_Toc196249198)

[Заключение 25](#_Toc196249199)

[Библиографический список 26](#_Toc196249200)

[ПРИЛОЖЕНИЕ. Листинг кода 27](#_Toc196249201)

# Введение

Целью курсового проекта является ...

Пользователями разрабатываемой системы ….

Объектом исследования являются ….

Предметом исследования в данной работе является ….

В качестве методов исследования используется анализ предметной области, исследование и моделирование процессов, разработка проекта с использованием UML-диаграмм, а также реализация системы на языке программирования высокого уровня.

Для моделирования информационной системы был использован структурный подход (нотация IDEF0 - интструмент Ramus) и объектно-ориентированный подход (нотация UML – интструмент Visual Paradigm). При реализации использовался язык программирования C# с использование сторонних библиотек и фреймворков. В качестве системы управления базами данных использовался ….

# Описание предметной области

….

(Вариант задания)

# Требования к проекту

**1. Назначение и цели создания системы**

Система «…» предназначена для обмена/хранения/извлечения и т.д.

Целью системы является … (оптимизация процессов/ улучшение показателей качества/ увелечение количества пользователей/уменьшение времени работы пользователя и т.д.)

**2. Требования к системе**

Требования к пользователям системы

….

* 1. Требование к функциям (задачам), выполняемым системой

| Функция | Исходные данные | Результат |
| --- | --- | --- |
| Клиентская часть (неавторизованный пользователь) | | |
| Регистрация | Запрос к серверу приложений для регистрации пользователя в системе  Запрос содержит следующие данные:  e-mail, имя пользователя, пароль, подтверждение пароль. | В случае корректных данных:  Ответ от сервера приложений, переход на страницу подтверждения регистрации.  В случае некорректных данных:  Ответ от сервера приложений,  с ошибкой регистрации и ее отображение на странице этой ошибке. |
| Авторизация | Запрос к серверу приложений для авторизации пользователя в системе  Запрос содержит следующие данные:  e-mail, пароль | В случае корректных данных:  Ответ от сервера приложений, переход на главную страницу приложения.  В случае некорректных данных:  Ответ от сервера приложений, с ошибкой и отображении ее на странице авторизации. |
| Клиентская часть (пользователь авторизованный) | | |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| Серверная часть (сервер-приложение)\* | | |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

\*если серверная часть есть

1. **Требования к атрибутам качества**

3.1 Требования к производительности

К количеству одновременно работающих пользователей системы на начальном этапе не должно ….

Количество одновременно выполняемых запросов к серверу не должно превышать ….

Время отклика сервера на входящие запросы не должно превышать … с при количестве одновременно работающих пользователей, не превышающем ...

Время отклика сервера на входящие запросы не должно превышать … с при количестве одновременно работающих пользователей, превышающем ...

3.2 Требования к безопасности

Требования не предъявляются / Предъявляются требования к …

3.3 Требования к эргономике и технической эстетике

Дизайн пользовательского интерфейса клиентского приложения должен ….

3.4 Требования к защите информации от несанкционированного доступа

Для получения полного доступа к функциям Системы необходимо ….

Для надежной передачи данных …

3.5 Требования к доступности

* 1. Непрерывная работа системы. Система должна быть доступна 99% времени
  2. …

1. **Требования к сквозной функциональности**
   1. Обработка ошибок в системе. Должна быть предусмотрена обработка исключительных ситуаций для бесперебойной работы.
   2. Функция логирования ошибок системы (протоколирование) / логирование операций пользователя. Должна быть поддержка нескольких уровней логгирования, например, для ошибок – уровень «error», для отладочной информации – «debug».
   3. Валидация полей ввода в клиентском приложении. В системе должна быть предусмотрена проверка доступных полей на заполненность и соответствие различным ограничениям.
   4. Валидация данных приходящих на сервер. В системе должна быть предусмотрена проверка данных для записи в БД.
   5. Шифрование запросов …
   6. Кеширование ….
   7. ….

# Оценка архитектурного решения

Предоставим несколько потенциальных архитектурных решений и оценим их с помощью облегчённого метода оценки архитектур-альтернатив («Lightweight Architecture Alternative Assessment Method»).

**1. Построение дерева атрибутов качества.**

Для разрабатываемой системы можно выделить следующие категории и подкатегории атрибутов качества:

* *Доступность: непрерывная работа системы*
* *Производительность: Масштабируемость системы при высокой загрузке, быстрый отклик на запрос пользователя*
* *Расширяемость: простота добавления нового функционала*
* *Сложность реализации: начальная разработка архитектуры проекта*

Атрибуты качества должны соответствовать   
пункту 3 в разделе Требования к проекту

**2. Ранжирование узлов дерева.**

На данном шаге необходимо определить приоритет каждой категории и подкатегории атрибутов качества и на основе полученных рангов распределить веса узлов дерева. Чем больше вес узла, тем приоритетнее данный атрибут качества для разрабатываемой системы.

Расставляются приоритеты и присваиваются значения весов согласно таблице весов ранга:



Результат ранжирования представлен на рисунке 1.

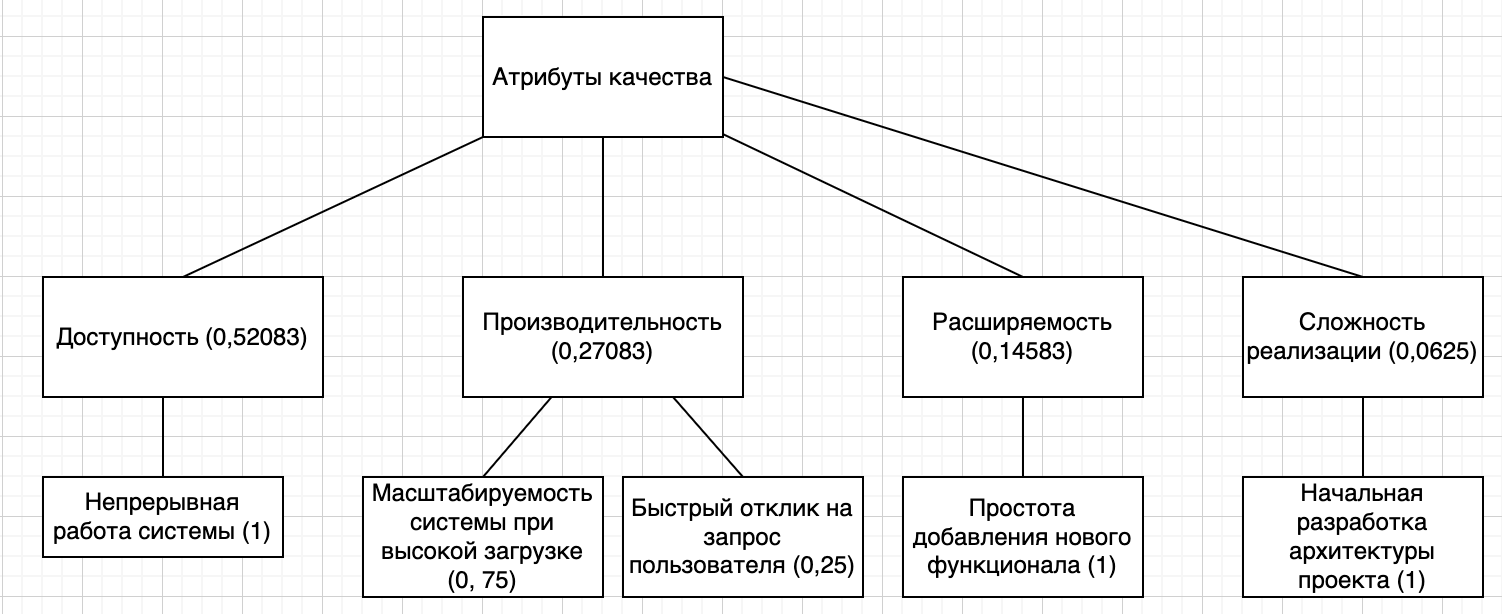


Рисунок 1. Атрибуты качества

**3. Предоставление архитектур-кандидатов.**

В качестве альтернативных решений будут рассматриваться следующие варианты архитектур:

а) Микросервисная архитектура

Микросервисная архитектура - это стиль проектирования программного обеспечения, где приложение разбито на маленькие независимые и взаимодействующие между собой компоненты - микросервисы. Каждый микросервис выполняет свою собственную задачу и может быть развернут, масштабирован, и заменен независимо от других микросервисов.

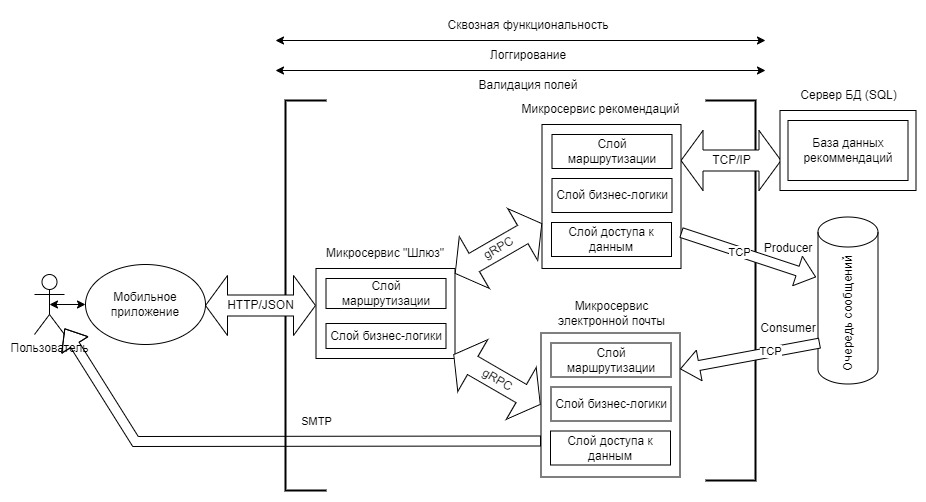
Преимущества:

* Гибкость: каждый микросервис может быть разработан независимо от других, что позволяет быстро адаптироваться к изменениям среды.
* Масштабируемость: каждый микросервис можно масштабировать отдельно, что позволяет увеличить производительность только тех компонентов, которые это требуют.
* Сопровождаемость: разные микросервисы можно сопровождать независимо друг от друга, что позволяет снизить затраты на обслуживание всей системы.
* Использование разных технологий: различные микросервисы могут быть разработаны на разных технологиях, что позволяет использовать наиболее эффективные инструменты для каждого компонента.

Недостатки:

* Сложность: микросервисная архитектура может быть достаточно сложной для понимания и управления, особенно если вы работаете над большой системой.
* Затраты на инфраструктуру: каждый микросервис требует инфраструктуры и ресурсов для работы, что может привести к увеличению затрат.
* Необходимость управления взаимодействием: необходимо управлять взаимодействием между различными микросервисами, чтобы обеспечить эффективную работу всей системы.
* Необходимость грамотного планирования: микросервисная архитектура требует грамотного планирования и дизайна, чтобы минимизировать потенциальные проблемы.

Данная система в реализации микросервисной архитектуры будет выглядеть следующим образом:

  
Рисунок 2. Микросервисная архитектура

б) Монолитная архитектура

Монолитная архитектура - это классический подход к проектированию приложений, когда весь код и логика хранятся в одном приложении, которое запускается на одном сервере.

Преимущества:

* Простота: монолит приложение относительно проще в разработке и управлении, чем система, состоящая из микросервисов.
* Легко масштабируемость: монолит можно развернуть на нескольких серверах и настроить балансировку нагрузки, что позволяет успешно масштабировать приложение.
* Удобное тестирование: для монолита тестирование происходит внутри единого процесса и необходимые связи между различными компонентами существуют уже на уровне программного кода.
* Простота развертывания: монолит легко развернуть на любом сервере, таким образом, минимизируя вероятность возникновения проблем.

Недостатки:

* Сложность масштабирования: монолит сложнее масштабировать, чем систему, разбитую на микросервисы.
* Высокий уровень связности: в монолите между компонентов может существовать тесная связь, что делает проект менее гибким и менее устойчивым к изменениям.
* Сложность внесения изменений: внесение изменений в одно компоненты может затронуть другие компоненты, что может привести к проблемам в работе всего приложения.
* Необходимость внесения изменений: монолит может стать слишком громоздким и сложно крупным для поддержки со временем, что может привести к большим затратам на разработку и поддержку.

Данная система в реализации монолитной архитектуры будет выглядеть следующим образом:

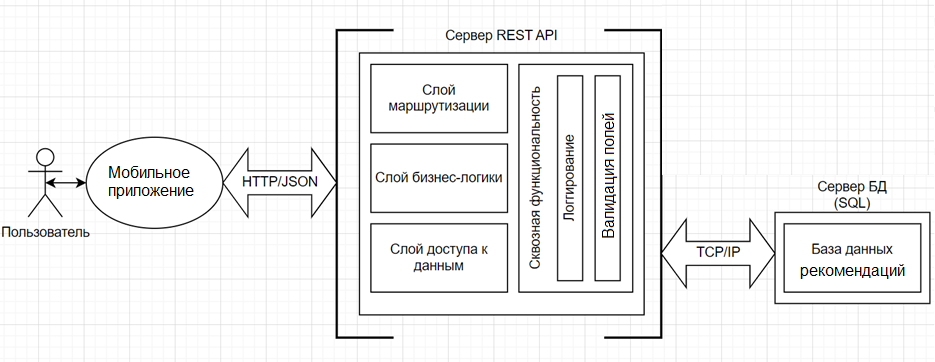


Рисунок 3. Монолитная архитектура

Сравним два решения согласно выделенным сценариям атрибутов качества системы.

**4. Оценка решений**

|  |  |  |  |
| --- | --- | --- | --- |
| **Сценарий** | **Вес** | **Архитектура 1** | **Архитектура 2** |
| 1.Непрерывная работа системы (99%) | 0,52083 | 4 | 4 |
| 2.1. Быстрый отклик на запрос пользователя до 2 секунд | 0,0677 | 4 | 4 |
| 2.2. Масштабируемость различных модулей системы при высокой загрузке | 0,2031 | 4 | 2 |
| 3. Простота добавления нового функционала | 0,14583 | 4 | 2 |
| 4. Начальная разработка архитектуры | 0,0625 | 2 | 4 |
| Итого |  | 3,6 | 3,2 |

**5. Анализ результатов**

По полученным результатам можно сделать вывод о том, что микро-сервисная архитектура является наиболее подходящим под имеющиеся нужды решением среди представленных вариантов.

Далее рассмотрим проектное решение на основе микросервисной архитектуры.

# Проектное решение

**Архитектурная модель**, используемая в данном приложении, представляет собой слоистую архитектуру. Она разбита на несколько слоев, каждый из которых отвечает за определенную функциональность, включая слой маршрутизации, слой бизнес-логики и слой работы с данными.

**Архитектурный стиль**, выбранный для данного приложения, является микросервисной архитектурой. Он предполагает, что приложение разделено на множество микросервисов, каждый из которых выполняет определенную задачу. Среди них можно выделить ....

**Паттерны и фреймворки**

MVVM

Приложение будет разрабатываться по шаблону Model-View-ViewModel.

…

Итогом применения паттерна MVVM является функциональное разделение приложения на три компонента, которые проще разрабатывать и тестировать, а также в дальнейшем модифицировать и поддерживать.

**IDE и языки программирования**

Язык программирования С#

Visual Studio - интегрированная среда разработки, стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений.

**Структурная модель проекта**

На рисунках … предоставлена контекстная диаграмма и декомпозиция в нотации IDEF0.

….

**Модель вариантов использования** **(Use Case Diagram)**

Данная диаграмма отражает отношения между актерами и прецедентами. Диаграмма является составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

На рисунке 1 изображена диаграмма вариантов использования для неавторизированного пользователя. Актером на данной диаграмме является …

Задачи … пользователя заключаются в …

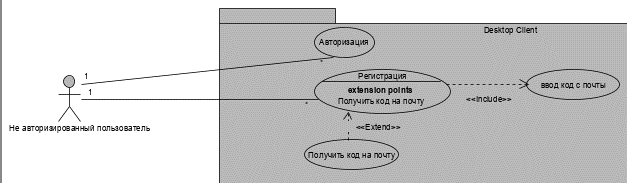


Рисунок 1. Диаграмма вариантов использования для неавторизированного пользователя.

На рисунке 2 изображена диаграмма, которая демонстрирует возможности … пользователя.

На данной диаграмме продемонстрированы следующие действия:

…

**Модель классов (Class Diagram)**

Данная диаграмма демонстрирует общую структуру иерархии классов системы, их коопераций, атрибутов, методов, интерфейсов и взаимосвязей между ними.

Для начала рассмотрим пакет, содержащий логику клиентского приложения.

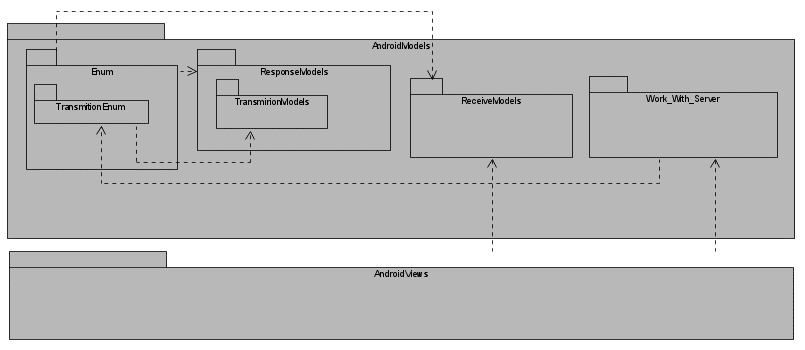


Рисунок 4. Диаграмма классов. Пакет логики клиентского приложения.

На диаграмме представлены 2 пакета. AndroidView – пакет содержащий логику форм и разметку. AndroidModel – пакет содержащий модели для передачи и принятия данных от сервера, также содержит подключение к серверу.

Рассмотрим пакет AndroidView.

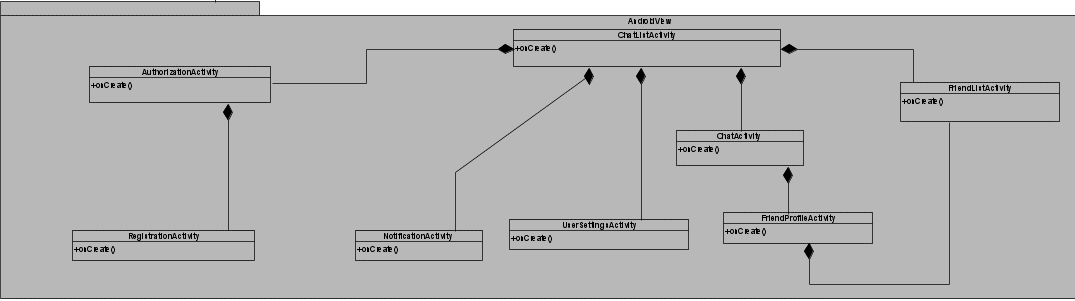


Рисунок 5. Диаграмма классов. Пакет AndroidView

На данной диаграмме представлены окна и методы реализации логики взаимодействия с API.

Рассмотрим подробнее каждое окно:

1. AuthorizationActivity, RegistrationActivity – окно авторизации и регистрации пользователя.
2. ChatListActivity – пользовательский элемент на котором отображается список чатов, создание чатов и отправка сообщений.
3. ChatActivity – пользовательский элемент на котором отображается список сообщения.
4. FriendListActivity – пользовательский элемент на котором отображается список друзей и список пользователей зарегистрированных в системе.
5. FriendProfileActivity – пользовательский элемент, на котором отображается профиль пользователя.
6. NotificationActivity – пользовательский элемент на котором отображаются уведомления пользователя.
7. UserSettingsActivity – пользовательский элемент на котором отображается настройки профиля.

Логика взаимодействия с API заключается в событиях нажатия кнопок, отображения списков данных, взаимодействия между пользовательскими элементами, посредством передачи данных между ними.

Рассмотрим пакет AndroidModel.

…

**Модель последовательности действий (Sequence Diagram)**

… (при наличии диаграммы)

Рассмотрим на единой временной оси жизненный цикл заявленной сквозной функциональности.

….

**Модель состояния (State Diagram)**

… (при наличии диаграммы)

…

**Модель компонентов (Component Diagram)**

**….**

**Модель развёртывания (Deployment Diagram)**

…

**Генерация кода по проекту**

**(пункт заполняется при наличии сгенерированного кода!!!)**

Генерация кода была выполнена по моделям в пакете ClientServerToApi. Пример генерации кода:

…

Отличие сгенерированного кода по диаграмме классов и кодом в программе при реализации заключается в:

1. В сгенерированном коде зависимости между классами прописаны жестко.
2. В сгенерированном коде показаны поля класса, а нужно было свойства для облегчения доступа к данным.
3. …

# Реализация приложения

**Описание технологий разработки приложения**

Язык программирования – …

Используемая СУБД – … Для взаимодействия с СУБД средствами языка программирования используется библиотека ... Для автоматического управления SQL-миграциями используется менеджер миграций из библиотеки …

Для логгирования средствами языка программирования …

Для генерации отчётов в формате PDF используется библиотека …

**Демонстрация приложения**

В качестве демонстрации работы приложения приводится основной сценарий работы по контрольному примеру.

Описание основных экранов программы. На рис. … предоставлено …

**…**

**…**

**…**

**Диаграмма классов, сгенерированная по коду**

На рис. … предоставлен пакет/классы и т.д.

…

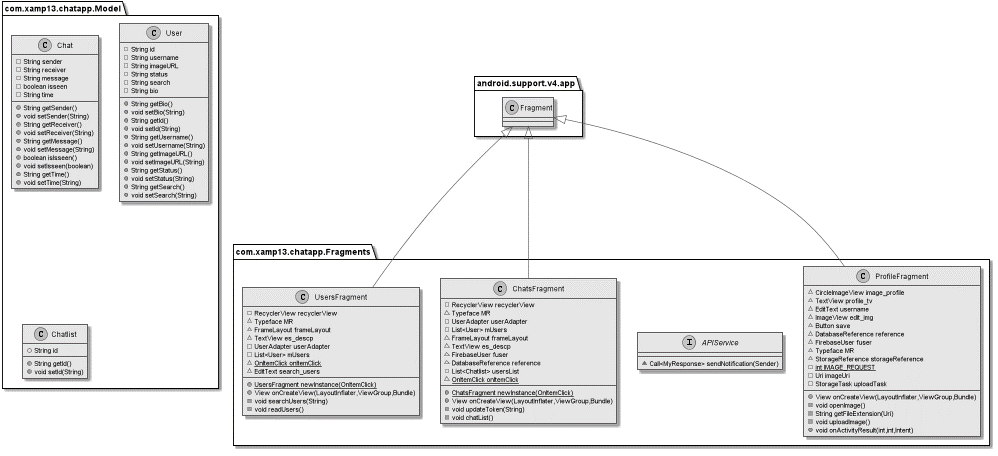


Рисунок ... Диаграмма классов. Пакет …

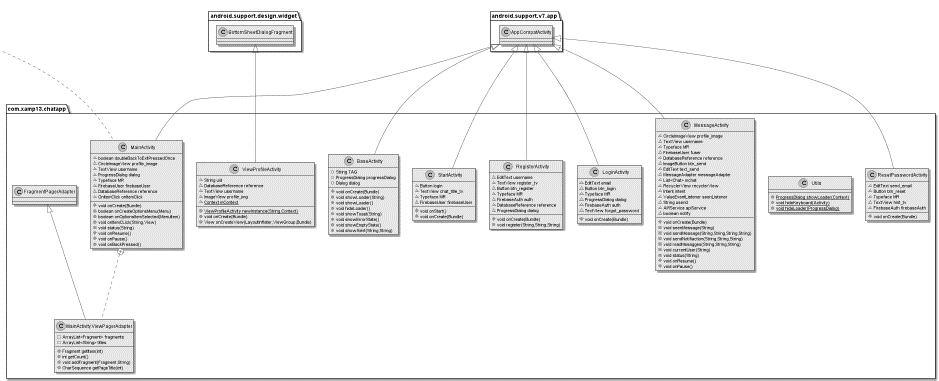


Рисунок ... Диаграмма классов. Пакет …

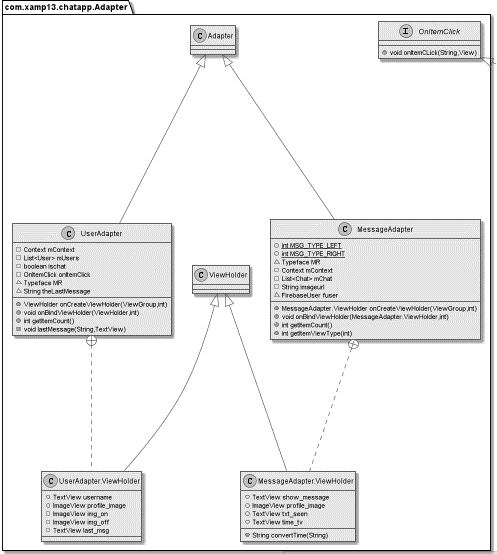


Рисунок ... Диаграмма классов. Пакет …

Сквозная функциональность в виде шифрования запросов к серверу и расшифровки ответов реализована в классе «Аdapter» на представленной выше диаграмме. Данная функция реализована в виде алгоритма шифрования AES, который использует 128-ми битный закрытый ключ.

Сквозная функциональность валидация и аутентификация реализована в классах «LoginActivity» и «RegisterActivity» в виде проверки вводимых пользователем данных перед отправкой запроса на сервер.

Логирование/протоколирование реализовано во всех классах, из представленных на диаграммах выше, экранов взаимодействия с пользователем для отслеживания его действий, а также при отправке запроса и получения ответ с сервера. Данные протоколы и логи ведутся для упрощения отладки возможных ошибок.

Обработка ошибок так же реализована в классах взаимодействия с пользователем и работы с сервером для предотвращения ситуаций непредвиденного прекращения работы приложения.

# Выводы по реализации приложения на основании проекта

Во время реализации проекта конечный вид проекта претерпел изменения, т.к. …. Изменились технологии реализации/ решено было применить … технологии/ …. / была измененая платформа разработки на .NET Xamarin и т.д.

В связись с этим произошли серьезные изменениния в функционале – это удаление … функций/ измение интрефейса/ …

Изменения, которые произошли с проектом в ходе разработки можно назвать существенными, так как оказали некоторое влияние на общую архитектуру системы. Но при всем этом, итоговая версия приложения отвечает всем основным требованиям, указанным в требованиях к проекту, и способна выполнять ключевые функции.

# Заключение

В результате был получен программный продукт, в котором были реализованы … поставленные цели и задачи – …

…

В ходе разработки системы получен опыт работы …. Так же были получены знания ….

Были получены навыки проектирования программных систем, что является неотъемлемым этапом в ходе разработки программных продуктов. Проектирование позволяет согласовать работу разработчиков, разделить зоны ответственности, продумать особенности реализации приложения на этапе проектного решения, предложить обоснованные варианты реализации, а также сложить общую картину ожидаемого результата.

# Библиографический список

1. Крючкова, Е. Н. Объектно-ориентированное программирование: Архитектурное проектирование и паттерны программирования : учебно-методическое пособие / Е. Н. Крючкова, С. М. Старолетов. — Барнаул : АлтГТУ, 2020. — 180 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/292790>
2. Флегонтов, А. В. Моделирование информационных систем. Unified Modeling Language : учебное пособие / А. В. Флегонтов, И. Ю. Матюшичев. — 2-е изд., стер. — Санкт-Петербург : Лань, 2022. — 112 с. — ISBN 978-5-8114-2907-3. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/206051
3. Корунова Н.В. Проектирование программного обеспечения: применение на практике / практикум по дисциплине «Проектирование информационных систем» / Н. В. Корунова. –Ульяновск : УлГТУ, 2019. – 69 с. - URL: […….](http://is.ulstu.ru/disc/pris). — Режим доступа: для авториз. пользователей.
4. Выполнение и оформление курсовых проектов (работ) [Текст]: методические указания / сост. Н. В. Корунова. - Ульяновск: УлГТУ, 2014. - 40 с. - Доступен также в Интернете <http://venec.ulstu.ru/lib/disk/2014/129.pdf>

….

# ПРИЛОЖЕНИЕ. Листинг кода

…