**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ДОНЕЦКОЙ НАРОДНОЙ РЕСПУБЛИКИ**

**ГОУВПО «ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**ФАКУЛЬТЕТ** Компьютерных наук и технологий\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**КАФЕДРА** Программной инженерии\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**НАПРАВЛЕНИЕ** 09.03.04 Программная инженерия\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ПРОФИЛЬ** Инженерия программного обеспечения\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Члены комиссии:

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

Подпись расшифровка подписи

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

Подпись расшифровка подписи

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

ППодпись расшифровка подписи

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту по дисциплине**

**«Архитектура и проектирование программного обеспечения»**

Руководитель

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

должность, звание подпись расшифровка подписи

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

должность, звание подпись расшифровка подписи

Студент

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

номер группы подпись расшифровка подписи

Донецк – 2021

РЕФЕРАТ

Тема курсового проекта: разработка информационной системы предприятия по установке и обслуживания газового оборудования.

В состав проекта входят: пояснительная записка 92 страницы, 13 рисунков, 29 таблиц, 3 приложения.

В курсовом проекте рассмотрен процесс проектирования программного обеспечения для системы предприятия по установке и обслуживания газового оборудования.

Проведен анализ предметной области и сформированы требования к разрабатываемой программной системе. Исходя из требований (атрибутов качества), выбрана архитектура системы с применением системных паттернов. Сделан обзор методологий разработки программного обеспечения, языков описания архитектур программных систем и применяемых видов. Выполнены этапы проектирования системы, включая разработку в CASE-среде различных моделей на языке UML.

С учётом типа приложения выбраны инструментальные средства разработки программного обеспечения Microsoft Visual Studio, веб-фреймворк Asp.Net. При разложении системы на объекты использованы паттерны проектирования. Создано программное обеспечение, обладающее следующими функциональными характеристиками: производительность, быстрая коммуникация с пользователями и надежность.

Область применения системы: предприятия, занимающиеся установкой и обслуживанием газового оборудования.

ПРОЕКТИРОВАНИЕ СИСТЕМЫ АВТОМАТИЗАЦИИ ДЕЯТЕЛЬНОСТИ ГАЗОВОГО ПРЕДПРИЯТИЯ, ПРОГРАММНАЯ СИСТЕМА, АРХИТЕКТУРА, МОДЕЛИ UML, ПАТЕРНЫ ПРОЕКТИРОВАНИЯ, ТЕХНОЛОГИЯ РАЗРАБОТКИ ПРОГРАММ

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc91486559)

[1 АНАЛИЗ ОБЪЕКТА АВТОМАТИЗАЦИИ 6](#_Toc91486560)

[1.1 Назначение системы и задачи автоматизации 6](#_Toc91486561)

[1.2 Формирование и анализ требований 6](#_Toc91486562)

[1.2.1 Бизнес-требования 6](#_Toc91486563)

[1.2.2 Требования пользователей 6](#_Toc91486564)

[1.2.3 Функциональные требования 7](#_Toc91486565)

[1.2.4 Системные требования 7](#_Toc91486566)

[1.2.5 Характеристика продукта 8](#_Toc91486567)

[1.2.6 Нефункциональные требования 9](#_Toc91486568)

[1.2.7 Атрибуты качества 9](#_Toc91486569)

[1.2.8 Анализ требования 10](#_Toc91486570)

[1.3 Задачи архитектуры программного обеспечения 29](#_Toc91486571)

[2 ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ СИСТЕМЫ 30](#_Toc91486572)

[2.1 Анализ ключевых вопросов проектирования 30](#_Toc91486573)

[2.1.1 Параллелизм 30](#_Toc91486574)

[2.1.2 Асинхронные агенты 30](#_Toc91486575)

[2.1.3 Обеспечение отказоустойчивости 31](#_Toc91486576)

[2.2 Стратегии и методы проектирования ПО 31](#_Toc91486577)

[2.3 Определение целей архитектуры 32](#_Toc91486578)

[2.4 Выявление основных сценариев 32](#_Toc91486579)

[2.5 Тип приложения 47](#_Toc91486580)

[2.6 Определение ограничений развёртывания 47](#_Toc91486581)

[2.7 Архитектурные стили проектирования 47](#_Toc91486582)

[2.8 Графическое представление архитектуры 48](#_Toc91486583)

[2.9 Детализация программ с помощью паттернов проектирования 53](#_Toc91486584)

[2.10 Пользовательский интерфейс 54](#_Toc91486585)

[2.11 Инструментальные средства разработки ПС. 55](#_Toc91486586)

[2.12 Программная реализация базовых модулей системы 56](#_Toc91486587)

[2.13 Анализ качества и оценка программного дизайна 58](#_Toc91486588)

[2.14 Анализ проектирования 60](#_Toc91486589)

[3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ МОДУЛЯ ПРОЕКТА 80](#_Toc91486590)

[3.1 Описание используемых паттернов 80](#_Toc91486591)

[3.2 Структура используемых паттернов 81](#_Toc91486592)

[3.3 Программный код паттернов 81](#_Toc91486593)

[ЗАКЛЮЧЕНИЕ 83](#_Toc91486594)

[СПИСОК ЛИТЕРАТУРЫ 84](#_Toc91486595)

[ПРИЛОЖЕНИЕ А Техническое задание 85](#_Toc91486596)

[ПРИЛОЖЕНИЕ Б Функциональная модель 89](#_Toc91486597)

[ПРИЛОЖЕНИЕ В Листринг программы 90](#_Toc91486598)

# ВВЕДЕНИЕ

Программы - это основной элемент повсеместно используемых сотовых телефонов, а также комплексных систем управления воздушным движением. Традиционные организации в финансовом, общественном и торговом секторах в значительной мере зависят от программного обеспечения. На сегодняшний день трудно найти организацию, которая хоть каким-либо образом не была вовлечена в сферу программного обеспечения.

Создание программного обеспечение требует широких и углубленных знаний в методологии программной инженерии, а также формировании умений и навыков проектирования программных систем с использованием различных моделей архитектурных решений.

Со временем программное обеспечение стало актуальным и в сфере предприятий, которые занимаются установкой газовым оборудованием. В связи с этим необходимо спроектировать архитектуру информационной системы предприятия по установке и обслуживанию газового оборудования с возможностью удаленного доступа с использованием различных архитектурных парадигм. Для этого необходимо спроектировать данную систему.

Результат процесса проектирования - проект (дизайн), включающий определение архитектуры, компонентов, интерфейсов и других характеристик программной системы или её компонентов, которые в совокупности являются основой для последующего конструирования программного продукта.

# 1 АНАЛИЗ ОБЪЕКТА АВТОМАТИЗАЦИИ

## Назначение системы и задачи автоматизации

Большинство людей тратят большое количество времени на поиск предприятия, которое занимается установкой газового оборудования и удовлетворяет личные требования заказчика. Пример таких требований может быть цена установки, время, место и условия работы. Это связано с тем, что большинство предприятий не имеет своей единой информационной системы, где можно ознакомиться с условиями работы и сравнить их с другими предприятиями. Из-за этого заказчику приходится тратить слишком много времени на поиск предприятия и обсуждение его условий работы.

Люди хотят иметь единый ресурс о предприятии, введя название услуги и город в интернете и узнав всю необходимую для себя информацию из ресурса, ознакомившись с информацией.

## Формирование и анализ требований

## 1.2.1 Бизнес-требования

1. Система должна автоматизировать процесс заказа услуг предприятия.
2. Система должна иметь единый информационный ресурс.

## 1.2.2 Требования пользователей

1. Система должна иметь возможность оплаты через различные платежные сервисы
2. Система должна быть удобна и понятная в использовании.
3. Система должна иметь функции помощи клиентам.
4. Система должна иметь защиту конфиденциальных данных.
5. Система должна иметь удобный поиск ближайших сервисов предприятий по месту жительства.

## 1.2.3 Функциональные требования

1. Система должна иметь доступ к актуальным услугам предприятия.
2. Система должна обладать онлайн консультациями клиентов
3. Система должна иметь возможность просматривать отзывы о предприятии
4. Система должна иметь возможность оставлять отзывы после проделанной услуги.
5. Система должна обладать видео демонстрацией процесса работы.
6. Система должна уметь уведомлять клиентов об изменении услуг.
7. Система должна иметь наличие чата для менеджмента, занимающегося консультацией клиентов.
8. Система должна иметь удобный инструментарий для создания расписаний.
9. Система должна уметь формировать скидки на услуги для всей сети предприятия.
10. Система должна предоставлять информацию о квалификации сотрудников выбранного сервиса предприятия.

## 1.2.4 Системные требования

1. Система должна корректно функционировать на IBM – совместимых PC со следующей системной конфигурацией:

- процессор Intel Core 2 Duo 1.3 GHz;

- оперативная память 2 Гб;

- дисковое пространство объёмом 3 гб;

- видеопамять 1 Гб.

## 1.2.5 Характеристика продукта

1. Система должна использовать распространённую распределённую систему базами данных Postgres SQL.
2. Система должна использовать клиент-серверную архитектуру.
3. Система должна предоставлять возможность оформления заказов на подключение и произведения оплаты услуг через Интернет.
4. Система должна обеспечивать гибкость системы.
5. Система должна использовать **SSH сертификат.**
6. **Система должна иметь** поддержку основных функций приложения СУБД, включающие в себя: формирование и редактирование таблиц базы данных расчетной системы; выставление счетов и их печать; составление отчетов; архивацию.
7. **Система** должна иметь базу, хранящую историю платежей, с обеспечением контроля сведений об оплате услуг.
8. **Система** должна иметь поддержку операции управления сетевым оборудованием: функции активации/деактивации доступа IP (блокировки/разблокировки).
9. В памяти системы должны храниться данные о клиентах, заключенных контрактах со сторонними поставщиками услуг, а также о стоимости передачи информации по разным каналам и направлениям.
10. Увеличение возможностей системы должно достигаться за счет модернизации аппаратной части системы.
11. СУБД должна быть совместима с различными компьютерными платформами, чтобы обеспечивать поддержку многопроцессорного режима работы.
12. Система должна иметь модульность, наличие таких подсистем как подсистема предварительной обработки данных, подсистема оперативного управления.

## 1.2.6 Нефункциональные требования

1. Должна использовать интуитивно понятный интерфейс.
2. Система должна иметь службу поддержки.
3. Система должна иметь систему подсказок при заполнении форм данных.
4. Требуется API-интерфейс для облегчения работы программистам.

## 1.2.7 Атрибуты качества

1. Система самостоятельно должна обрабатывать чрезвычайные ситуации.
2. Система должна записывать ошибки в логгер и файл, для дальнейшего обследования проблемы.
3. Система должна работать исключительно как информационный портал.
4. Система должна иметь подсистему оповещения на случай взлома хакерами.
5. Система должна уведомлять администрацию о нехватке места на сервере.
6. Система должна отображаться на ПК и телефонах.
7. Система должна сама распределять процессы обработки данных.
8. Система должна автоматически собирать отчёты по использованию
9. В системе должна быть защита от взлома базы данных с помощью объектно-реляционного отображения.
10. Система должна реализовывать асинхронную работу с БД.
11. Система должна предоставлять уникальный ключ оплаты услуги.
12. Система должна облегчить оплату услуг.
13. Система должна обладать высокой точностью при работе с деньгами пользователей.
14. При разработке нужно применить новейшие технологии в сфере веб программирования.
15. Требуется создать легковесную и быструю систему для оптимизации ресурса.
16. Система должна отображать список всех сервисов предприятия.
17. Система должна правильно обрабатывать платежи.
18. Система должна обеспечивать защиту конфиденциальных данных верификации пользователя.
19. Система должна иметь возможность анонимных опросов пользователей.
20. Система должна иметь прокси сервер.
21. Система должна поддерживать функцию резервного копирования данных.
22. Система должна поддерживать экспорт базы данных в форматы csv, xlsx, json, xml.
23. Система должна иметь защиту от ботов и роботов.
24. Настройка аппаратного комплекса и системного ПО для реализации системы должна быть выполнена в соответствии со всеми действующими нормами по обеспечению доступности для информационной инфраструктуры заказчика и интегрирована в среду резервного копирования заказчика.

## 1.2.8 Анализ требования

Анализ требований представлен в таблицах 1.1-1.6.

Таблица 1.1 - Оценочные элементы фактора "Надежность ПС"

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| Н0101 | Наличие требований к программе по устойчивости функционирования при наличии ошибок во входных данных | Экспертный | 1 |
| Н0102 | Возможность обработки ошибочных ситуаций | То же | 1 |
| Н0103 | Полнота обработки ошибочных ситуаций | " | 0,8 |
| Н0104 | Наличие тестов для проверки допустимых значений входных данных |  | 1 |
| Н0105 | Наличие системы контроля полноты входных данных |  | 1 |
| Н0106 | Наличие средств контроля корректности входных данных |  | 1 |
| Н0107 | Наличие средств контроля непротиворечивости входных данных |  | 1 |
| Н0109 | Наличие обработки граничных результатов |  | 1 |
| Н0110 | Наличие обработки неопределенностей (указание одного гражданства с предъявлением паспорта другого). |  | 0.8 |

Продолжение таблицы 1.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Н0201 | Наличие требований к программе по восстановлению процесса выполнения в случае сбоя операционной системы, процессора, внешних устройств |  | 1 | |
| H0202 | Наличие требований к программе по восстановлению результатов при отказах процессора, ОС |  | 1 | |
| Н0203 | Наличие средств восстановления процесса в случае сбоев оборудования |  | 0,4 | |
| Н0204 | Наличие возможности разделения по времени выполнения отдельных функций программ |  | 1 |
| Н0205 | Наличие возможности повторного старта с точки останова |  | 0.5 |
| H0301 | Наличие централизованного управления процессами, конкурирующими из-за ресурсов |  | 0.7 |

Продолжение таблицы 1.1

|  |  |  |  |
| --- | --- | --- | --- |
| H0302 | Наличие возможности автоматически обходить ошибочные ситуации в процессе вычисления |  | 1 |
| Н0303 | Наличие средств, обеспечивающих завершение процесса решения в случае помех |  | 1 |
| H0304 | Наличие средств, обеспечивающих выполнение программы в сокращенном объеме в случае ошибок или помех |  | 0.8 |
| Н0305 | Показатель устойчивости к искажающим воздействиям | Расчетный | https://api.docs.cntd.ru/img/12/00/00/91/35/34ee1e2c-6c63-4430-8809-58fff8c40c7a/P011300530000.png, P(Y) = 1 – 47/50 = 0,96 |

Таблица 1.2 - Оценочные элементы фактора "Сопровождаемость"

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| C0101 | Наличие модульной схемы программы |  | 1 |
| C0102 | Оценка программы по числу уникальных модулей |  | 1 |
| C0201 | Наличие ограничений на размеры модуля |  | 1 |

Продолжение таблицы 1.2

|  |  |  |  |
| --- | --- | --- | --- |
| C0301 | Наличие проверки корректности передаваемых данных |  | 1 |
| С0302 | Оценка простоты программы по числу точек входа и выхода | Расчетный | https://api.docs.cntd.ru/img/12/00/00/91/35/34ee1e2c-6c63-4430-8809-58fff8c40c7a/P0117000B0000.png, = |
| С0303 | Осуществляется ли передача результатов работы модуля через вызывающий его модуль | Экспертный | 1 |
| С0304 | Осуществляется ли контроль за правильностью данных, поступающих в вызывающий модуль от вызываемого |  | 0,7 |
| С030 | Наличие требований к независимости модулей программы от типов и форматов выходных данных |  | 1 |

Таблица 1.3 - Оценочные элементы фактора "Удобство применения"

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| У0801 | Уровень языка общения пользователя с программой |  | 1 |

Продолжение таблицы 1.3

|  |  |  |  |
| --- | --- | --- | --- |
| У0802 | Легкость и быстрота загрузки и запуска программы |  | 0,9 |
| У0803 | Легкость и быстрота завершения работы программы |  | 0,2 |
| У0804 | Возможность распечатки содержимого чека заказа |  | 1 |
| У0805 | Возможность приостанова и повторного запуска работы без потерь информации |  | 1 |
| У0901 | Соответствие меню требованиям пользователя |  | 0.6 |
| У0902 | Возможность прямого перехода вверх и вниз по многоуровнему меню (пропуск уровней) |  | 1 |
| У1001 | Возможность управления подробностью получаемых выходных данных |  | 0.8 |
| У1002 | Достаточность полученной информации для продолжения работы |  | 1 |

Продолжение таблицы 1.3

|  |  |  |  |
| --- | --- | --- | --- |
| У1101 | Обеспечение удобства ввода данных |  | 1 |
| У1102 | Легкость восприятия |  | 1 |
| У1201 | Обеспечение программой выполнения предусмотренных рабочих процедур |  | 1 |
| У1202 | Достаточность информации, выдаваемой программой для составления дополнительных процедур |  | 1 |

Таблица 1.4 - Оценочные элементы фактора "Эффективность"

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| Э0101 | Проблемно-ориентированные функции | Экспертный или расчетный | 1 |
| Э0102 | Машинно-ориентированные функции | То же | 0,6 |
| Э0103 | Функции ведения и управления |  | 1 |
| Э0104 | Функции ввода/вывода |  | 1 |
| Э0105 | Функции защиты и проверки данных |  | 1 |
| Э0106 | Функции защиты от несанкционированного доступа |  | 1 |
| Э0107 | Функции контроля доступа |  | 1 |
| Э0108 | Функции защиты от внесения изменений |  | 1 |

Продолжение таблицы 1.4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Э0109 | Наличие соответствующих границ функциональных областей | | |  | | 1 |
| Э0110 | Число знаков после запятой в результатах вычислений | | |  | | 0,9 |
| Э0201 | Время выполнения программ | | |  | | 0,7 |
| Э0202 | Время реакции и ответов | | |  | | 1 |
| Э0203 | Время подготовки | | |  | | 1 |
| Э0205 | Затраты времени на защиту данных | | |  | | 1 |
| Э0206 | | Время компиляции |  | | 0.6 | |
| Э0301 | | Требуемый объем внутренней памяти |  | | 0.7 | |
| Э0302 | | Требуемый объем внешней памяти |  | | 1 | |
| Э0303 | | Требуемые периферийные устройства |  | | 1 | |
| Э0304 | | Требуемое базовое программное обеспечение |  | | 1 | |

Таблица 1.5 - Оценочные элементы фактора "Универсальность"

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| Г0101 | Оценка числа потенциальных пользователей | Экспертный | 1 |
| Г0102 | Оценка числа функций ПС | То же | 1 |
| Г0103 | Насколько набор функций удовлетворяет требованиям пользователя |  | 1 |

Продолжение таблицы 1.5

|  |  |  |  |
| --- | --- | --- | --- |
| Г0104 | Насколько возможности программ охватывают область решаемых пользователем задач |  | 1 |
| Г0105 | Возможность настройки формата выходных данных для конкретных пользователей |  | 1 |
| Г0201 | Наличие схемы иерархии модулей программы |  | 1 |
| Г0202 | Оценка независимости модулей |  | 0,8 |
| Г0203 | Оценка числа уникальных элементов/реквизитов |  | 1 |
| Г0204 | Используется ли в текущем вызове модуля информация, полученная в предыдущем вызове |  | 0.5 |
| Г0205 | Оценка организации точек входа и выхода модуля |  | 1 |
| Г0206 | Наличие описания атрибутов модуля |  | 1 |
| Г0301 | Оценка программ по числу переходов и точек ветвления |  | 1 |
| Г0701 | Оценка зависимости программ от емкости оперативной памяти ЭВМ |  | 0.6 |

Продолжение таблицы 1.5

|  |  |  |  |
| --- | --- | --- | --- |
| Г0702 | Оценка зависимости временных характеристик программы от скорости вычислений ЭВМ |  | 1 |
| Г0703 | Оценка зависимости функционирования программы от числа внешних запоминающих устройств и их общей емкости |  | 1 |
| Г0704 | Оценка зависимости функционирования программы от специальных устройств ввода-вывода |  | 0.3 |
| Г0801 | Применение специальных языков программирования |  | 1 |
| Г0802 | Оценка зависимости программы от программ операционной системы |  | 0.5 |
| Г0803 | Зависимость от других программных средств |  | 0.5 |
| Г0901 | Оценка локализации непереносимой части программы |  | 1 |

Таблица 1.6 - Оценочные элементы фактора "Корректность"

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| К0101 | Наличие всех необходимых документов для понимания и использования ПС | Экспертный | 1 |

Продолжение таблицы 1.6

|  |  |  |  |
| --- | --- | --- | --- |
| К0102 | Наличие описания и схемы иерархии модулей программы |  | 1 |
| К0103 | Наличие описания основных функций |  | 1 |
| К0104 | Наличие описания частных функций |  | 1 |
| К0105 | Наличие описания данных |  | 1 |
| К0106 | Наличие описания алгоритмов |  | 1 |
| К0107 | Наличие описания интерфейсов между модулями |  | 1 |
| К0108 | Наличие описания интерфейсов с пользователями |  | 1 |
| К0109 | Наличие описания используемых числовых методов |  | 1 |
| К0110 | Указаны ли все численные методы |  | 1 |
| К0111 | Наличие описания всех параметров |  | 1 |
| К0112 | Наличие описания методов настройки системы |  | 1 |
| К0113 | Наличие описания всех диагностических сообщений |  | 1 |
| К0114 | Наличие описания способов проверки работоспособности программы |  | 1 |
| К0301 | Отсутствие противоречий в описании частных функций |  | 1 |

Продолжение таблицы 1.6

|  |  |  |  |
| --- | --- | --- | --- |
| К0302 | Отсутствие противоречий в описании основных функций в разных документах |  | 1 |
| К0303 | Отсутствие противоречий в описании алгоритмов |  | 1 |
| К0304 | Отсутствие противоречий в описании взаимосвязей в системе |  | 1 |
| К0305 | Отсутствие противоречий в описании интерфейсов между модулями |  | 1 |
| К0306 | Отсутствие противоречий в описании интерфейсов с пользователем |  | 1 |
| К0307 | Отсутствие противоречий в описании настройки системы |  | 1 |
| К0309 | Отсутствие противоречий в описании иерархической структуры сообщений |  | 1 |
| К0310 | Отсутствие противоречий в описании диагностических сообщений |  | 1 |

Продолжение таблицы 1.6

|  |  |  |  |
| --- | --- | --- | --- |
| К0311 | Отсутствие противоречий в описании данных |  | 1 |
| К0501 | Единообразие способов вызова модулей |  | 1 |
| К0502 | Единообразие процедур возврата управления из модулей |  | 1 |
| К0503 | Единообразие способов сохранения информации для возврата |  | 1 |
| К0504 | Единообразие способов восстановления информации для возврата |  | 1 |
| К0505 | Единообразие организации списков передаваемых параметров |  | 1 |
| К0601 | Единообразие наименования каждой переменной и константы |  | 1 |
| К0602 | Все ли одинаковые константы встречаются во всех программах по одинаковыми именами |  | 1 |
| К0603 | Единообразие определения внешних данных во всех программах |  | 1 |

Продолжение таблицы 1.6

|  |  |  |  |
| --- | --- | --- | --- |
| К0604 | Используются ли разные идентификаторы для разных переменных |  | 1 |
| К0605 | Все ли общие переменные объявлены как общие переменные |  | 1 |
| К0606 | Наличие определений одинаковых атрибутов |  | 1 |
| К0701 | Комплектность документации в соответствии со стандартами |  | 1 |
| К0702 | Правильное оформление частей документов |  | 1 |
| К0703 | Правильное оформление титульных и заглавных листов документов |  | 1 |
| К0704 | Наличие в документах всех разделов соответствии со стандартами |  | 1 |
| К0705 | Полнота содержания разделов в соответствии со стандартами |  | 1 |

Продолжение таблицы 1.6

|  |  |  |  |
| --- | --- | --- | --- |
| К0706 | Деление документов на структурные элементы: разделы, подразделы, пункты, подпункты |  | 1 |
| К0801 | Соответствии организации и вычислительного процесса эксплуатационной документации |  | 1 |
| К0802 | Правильность заданий на выполнение программы, правильность написания управляющие и операторов (отсутствие ошибок) |  | 1 |
| К0803 | Отсутствие ошибок в описании действий пользователя |  | 1 |
| К0804 | Отсутствие ошибок в описании запуска |  | 1 |
| К0805 | Отсутствие ошибок в описании генерации |  | 1 |
| К0806 | Отсутствие ошибок в описании настройки |  | 1 |

Продолжение таблицы 1.6

|  |  |  |  |
| --- | --- | --- | --- |
| К1001 | Наличие требований к тестированию программ |  | 1 |
| К1002 | Достаточность требований к тестированию программ |  | 1 |
| К1003 | Отношение числа модулей, отработавших в процессе тестирования и отладки () к общему числу модулей () | Расчетный | 1 |
| К1004 | Отношение числа логических блоков, отработавших в процессе тестирования и отладки (), к общему числу логических блоков в программе () | То же | 1 |

Расчёт фактора «Надёжность»

*Q=10*

*Q=5*

*=0,7*

*=0,3*

Оценка фактора «Корректность»

Критерий «Полнота реализации»

*Q=14*

P1=М1=1.

Критерий «Согласованность»

Определение

*Q=5*

Определение

*Q=6*

Определение

*Q=6*

*=0.35*

*=0.35*

*=0.3*

Критерий «Проверенность»

*Q=2*

*=0.2*

*=0.38*

*=0.42*

=0.42\*1.1+0.38\*1.1+0.2\*1.1= 1.1

Вывод: разработчик ПС обеспечил установленный уровень качества выполнения этапа ЖЦ «Анализ требований» по надёжности () и по корректности ().

## 1.3 Задачи архитектуры программного обеспечения

Бизнес-цель 1. Увеличить скорость получения информации клиентам о предприятии.

Бизнес-цель-2. Увеличить узнаваемость компании.

Бизнес-цель-3. Увеличить скорость работы с клиентами, тем самым увеличив прибыль и расширяемость.

Критерий успеха-1*.* Аналогичный ресурс используют менее 20% предприятий в мире.

# 2 ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ СИСТЕМЫ

## 2.1 Анализ ключевых вопросов проектирования

## 2.1.1 Параллелизм

Параллельные вычисления — это свойство систем, при котором несколько вычислений выполняются одновременно, и при этом, возможно, взаимодействуют друг с другом. Вычисления могут выполняться на нескольких [ядрах](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D1%8F%D0%B4%D0%B5%D1%80%D0%BD%D1%8B%D0%B9_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80) одного [чипа](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D1%85%D0%B5%D0%BC%D0%B0) с [вытесняющим разделением времени](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%82%D0%B5%D1%81%D0%BD%D1%8F%D1%8E%D1%89%D0%B0%D1%8F_%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE%D0%B7%D0%B0%D0%B4%D0%B0%D1%87%D0%BD%D0%BE%D1%81%D1%82%D1%8C) [потоков](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%82%D0%BE%D0%BA_%D0%B2%D1%8B%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F) на одном процессоре, либо выполняться на физически отдельных [процессорах](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80).

Существуют различные способы реализации параллельных вычислений. Например, каждый вычислительный процесс может быть реализован в виде процесса операционной системы (ОС) либо вычислительные процессы могут представлять собой набор потоков выполнения внутри одного процесса ОС. Параллельные программы могут физически исполняться либо последовательно на единственном процессоре, осуществляя по очереди шаги выполнения каждого вычислительного процесса, либо параллельно, выделяя каждому вычислительному процессу один или несколько процессоров (находящихся рядом или объединенных в компьютерную сеть).

## 2.1.2 Асинхронные агенты

Асинхронность позволяет вынести отдельные задачи из основного потока в специальные асинхронные методы или блоки кода. Особенно это актуально в графических программах, где продолжительные задачи могу блокировать интерфейс пользователя. И чтобы этого не произошло, нужно задействовать асинхронность.

Также асинхронность несет выгоды в веб-приложениях при обработке запросов от пользователей, при обращении к базам данных или сетевым ресурсам. При больших запросах к базе данных асинхронный метод просто уснет на время, пока не получит данные от БД, а основной поток сможет продолжить свою работу. В синхронном же приложении, если бы код получения данных находился в основном потоке, этот поток просто бы блокировался на время получения данных.

В ходе проектирования информационной системы предприятия по установке и обслуживанию газового оборудования были выделены следующие асинхронные процессы:

1) обработка запросов пользователей (обеспечивается на уровне архитектуры приложения);

2) добавление, удаление и загрузка в базе данных (обеспечивается на уровне архитектуры базы данных).

## 2.1.3 Обеспечение отказоустойчивости

Одной из задач проектирования является обеспечение корректной обработки ошибок.

В проекте используется существует два метода контроля ошибок:

- исключения *—* метод, позволяющий разработчикам получить наиболее подробную информацию по ошибкам и достаточно легко их локализовать, классифицировать и обрабатывать централизованно.

- коды ошибок — метод, широко использующийся при обработке большого потока данных, приоритетом ставится скорость обработки информации. При появлении ошибки, функции возвращают ее код.

## 2.2 Стратегии и методы проектирования ПО

В ходе проектирования курсового проекта было использовано объектно-ориентированное проектирование в объединении с традиционной стратегией «снизу-вверх». Главную роль в данной стратегии играют полиморфизм и инкапсуляция.

При планировании порядка разработки программных модулей использовались абстрагирование, использование паттернов.

## 2.3 Определение целей архитектуры

Задачей архитектуры является разработка новой системы автоматизации деятельности предприятия по установке и обслуживанию газового оборудования со следующими основными функциями:

Функция-1. Доступ к актуальным услугам предприятия.

Функция-2. Онлайн консультация клиентов.

Функция-3. Просмотр отзывов о предприятии.

Функция-4. Пред-заказ услуг.

Функция-5. Видео демонстрация процесса работы.

Функция-6. Уведомление клиентов о изменении услуг.

Функция-7. Наличие чата для менеджмента, занимающегося консультацией клиентов.

Функция-8. Удобный инструментарий для создания расписаний.

Функция-9. Формирование скидок на услуги для всей сети предприятия.

Функция-10. Предоставление информации о квалификации сотрудников предприятия.

## 2.4 Выявление основных сценариев

На рисунках 2.1-2.2 показаны диаграммы вариантов использования ИСПУГО в нотации языка UML.

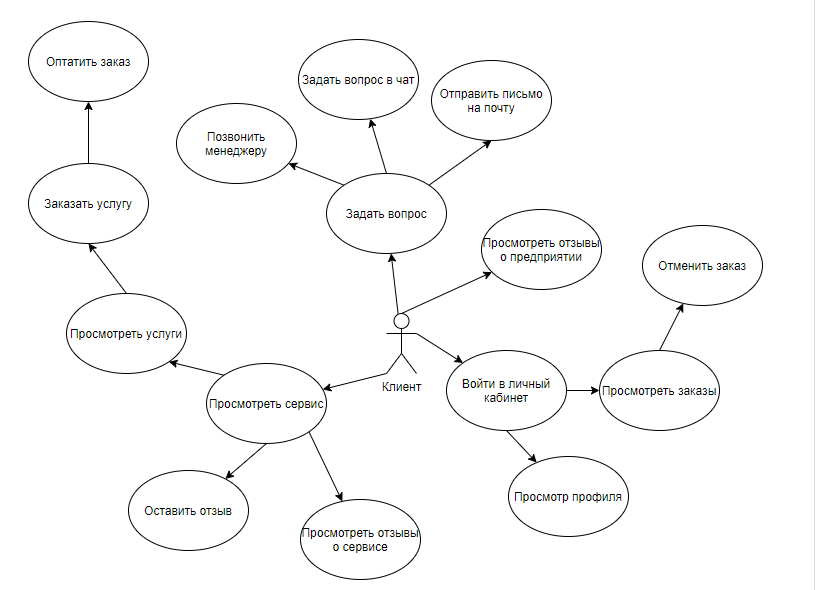


Рисунок 2.1 – Диаграмма прецедентов для действующего лица (актёра) Клиент

Формула и расчёты для диаграммы прецедентов для действующего лица (актёра) Клиент.

Диаграмма вошла в диапазон между 2.5 и 3, оценка допустима.

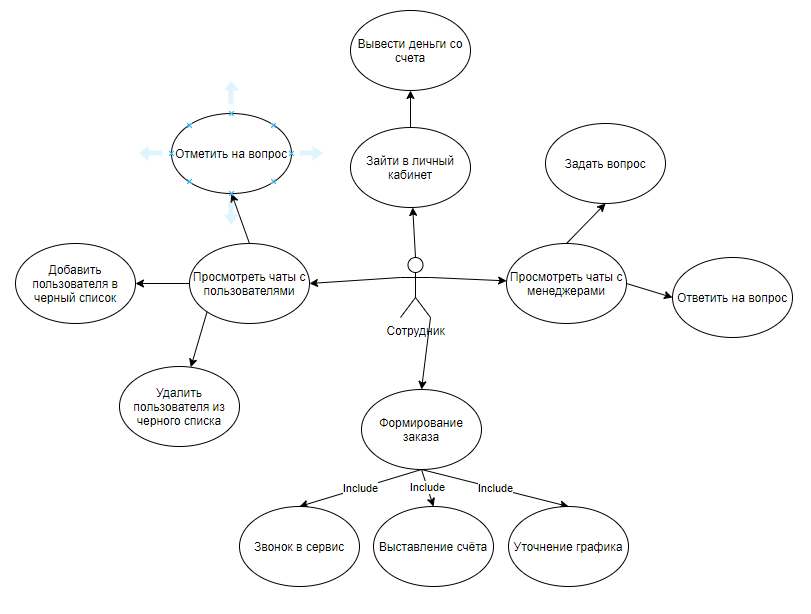


Рисунок 2.2 – Диаграмма прецедентов для действующего лица (актёра) Сотрудник системы

Формула и расчёты для диаграммы прецедентов для действующего лица (актёра) Сотрудник.

Диаграмма вошла в диапазон между 2.5 и 3, значит оценка допустима.

Ниже приведены спецификаций вариантов использования для ИСПУГО (см. табл. 2.1 – 2.14).

Таблица 2.1. Спецификация варианта использования «Просмотр отзывов о предприятии»

|  |  |
| --- | --- |
| **Название поля шаблона** | **Значение поля** |
| № прецедента | Вариант использования 1 |
| Название  прецедента | Просмотр отзывов о предприятии |
| Автор | Саевский О.В. |
| Дата создания | 02.12.2021 г. |
| Актёр | Клиент |
| Описание | Клиент запускает приложение ИСПУГО через интернет или с домашнего компьютера и просматривает отзывы о предприятии от других пользователей |
| Предусловия | - |
| Постусловия | 1. Пользователь не в черном списке. |
| Основной поток событий | 1.0. Открытие страницы отзывов предприятия.  1.Смотри общий рейтинг.  2.Смотрит отзывы о предприятии. |
| Альтернативные потоки событий | 1. Открытие страницы отзывов.  2. Просмотр отзывов о предприятии как большой сети. |

Продолжение таблицы 2.1

|  |  |
| --- | --- |
| Исключения | - |
| Включение | Нет |
| Приоритет | Средний |
| Частота  использования | Приблизительно 60000 пользователей в день. |
| Бизнес-правила | Бизнес-правило-1, Бизнес-правило-3 |
| Специальные требования | - |
| Предположения | 1.Предполагается, что 80% клиентов , которые будут в будущем заказывать услугу перед этим просмотрят отзывы |
| Замечания и вопросы | - |

Таблица 2.2. Спецификация варианта использования «Заказ услуги»

|  |  |
| --- | --- |
| **Название поля шаблона** | **Значение поля** |
| Название  прецедента | Заказ услуги |
| Автор | Саевский О.В. |
| Дата создания | 23.10.2021 г. |
| Актёр | Клиент |
| Описание | Клиент запускает приложение ИСПУГО через интернет или с домашнего компьютера и осуществляет заказ услуги в одном из сервисов. |
| Предусловия | 1. Клиент зарегистрирован в ИСПУГО. 2. Расписание данного сервиса должно быть свободным в данное время. |
| Постусловия | 2.Существует пользовательская верификация, а также специальный код заказа. |
| Основной поток событий | 1. Заказ услуги.   1. Клиент выбирает сервис.  2.Клиент выбирает услугу.  3.В данное время сервис не может принять заказ.  4.Смотрит предложения другого сервиса.  5.Если сервис свободен в данное время, то оформляет заказ. |
| Альтернативные потоки событий | 1. Заказ нескольких услуг. |

Продолжение таблицы 2.2

|  |  |
| --- | --- |
| Исключения | 1. Пользователи из черного списка не могут заказать услугу. |
| Включение | Нет |
| Приоритет | Самый высокий |
| Частота  использования | Приблизительно 3000 пользователей за день. |
| Бизнес-правила | Бизнес-правило-3, Бизнес-правило-1. |
| Специальные требования | 1.Клиент должен иметь возможность отменить заказ.  2.Клиент должен иметь возможность просматривать все активные заказы. |
| Предположения | Все сервисы заняты в данное время. |
| Замечания и вопросы | 1.Требуется доп. оплата за срочный заказ. |

Таблица 2.3. Спецификация варианта использования «Задать вопрос в чат»

|  |  |
| --- | --- |
| **Название поля шаблона** | **Значение поля** |
| Название  прецедента | Задать вопрос в чат |
| Автор | Саевский О.В. |
| Дата создания | 02.12.2021 г. |
| Актёр | Клиент |
| Описание | Клиент запускает приложение ИСПУГО через интернет или с домашнего компьютера и задает вопрос в чат. |
| Предусловия | Клиент зарегистрирован в ИСПУГО. |
| Постусловия | Нет. |
| Основной поток событий | 1.0. Пользователь задает вопрос в чате.  1.Клиент открывает чат.  2.Пишет туда свой вопрос. |
| Альтернативные  потоки событий | Альтернативного потока событий нет. |
| Исключения | 1.0. Пользователь находится в черном списке. |
| Включение | Нет |
| Приоритет | Высокий |
| Частота  использования | Приблизительно 1000 раз в день. |
| Бизнес-правила | Бизнес-правило-1 |
| Специальные  требования | Специальных требований нет |
| Предположения | 1.Клиент зарегистрирован в приложении. |

Продолжение таблицы 2.3

|  |  |
| --- | --- |
| Замечания и  вопросы | 1.Система сохраняет историю переписки и на её основе формирует отчет о часто задаваемых вопросах. |

Таблица 2.4. Спецификация варианта использования «Просмотреть услуги»

|  |  |
| --- | --- |
| **Название поля шаблона** | **Значение поля** |
| Название  прецедента | Просмотреть услуги |
| Автор | Саевский О.В. |
| Дата создания | 02.12.2021 г. |
| Актёр | Клиент |
| Описание | Клиент запускает приложение ИСПУГО через интернет или с домашнего компьютера и осуществляет просмотр услуг. |
| Предусловия | Нет |
| Постусловия | Нет |
| Основной поток событий | 1.0. Просмотр услуг  1.Клиент выбирает местоположение сервиса.  2.Просматривает услуги, которые он предоставляет. |
| Альтернативные  потоки событий | Альтернативного потока событий нет |
| Исключения | Нет |
| Включение | Нет |
| Приоритет | Самый высокий. |
| Частота  использования | Приблизительно 90000 раз в день. |
| Бизнес-правила | Бизнес-правило-1, Бизнес-правило-2, Бизнес-правило-3 |
| Специальные  требования | Специальных требований нет |
| Предположения | Клиенту ищет конкретную услугу. |
| Замечания и  вопросы | Замечаний нет. |

Таблица 2.5. Спецификация варианта использования «Оставить отзыв»

|  |  |
| --- | --- |
| **Название поля шаблона** | **Значение поля** |
| Название  прецедента | Оставить отзыв |
| Автор | Саевский О.В. |
| Дата создания | 02.12.2021 г. |
| Актёр | Клиент |

Продолжение таблицы 2.5

|  |  |
| --- | --- |
| Описание | Клиент запускает приложение ИСПУГО через интернет или с домашнего компьютера и оставляет отзыв о сервисе. |
| Предусловия | Клиент зарегистрирован в ИСПУГО.  Клиент выбрал конкретный сервис.  Клиент не в черном списке. |
| Постусловия | Клиент зарегистрирован в приложении. |
| Основной поток событий | 1.0. Клиент оставляет отзыв о сервисе.  1.Клиент осуществляет вход в систему.  2.Выбирает конкретный сервис.  3.Оставляет отзыв. |
| Альтернативные  потоки событий | Альтернативного потока событий нет |
| Исключения | 1.0. Пользователь в черном списке. |
| Включение | Нет |
| Приоритет | Высокий |
| Частота  использования | Приблизительно 1000 раз в день. |
| Бизнес-правила | Бизнес-правило-3, Бизнес-правило-1 |
| Специальные  требования | Нет |
| Предположения | Клиент зарегистрирован. |
| Замечания и  вопросы | Замечаний нет. |

Таблица 2.6. Спецификация варианта использования «Оказание помощи клиенту»

|  |  |
| --- | --- |
| **Название поля шаблона** | **Значение поля** |
| Название  прецедента | Оказание помощи клиенту |
| Автор | Саевский О.В. |
| Дата создания | 02.12.2021 г. |
| Актёр | Сотрудник |
| Описание | У клиента возник вопрос, который он не может решить без помощи специалиста. |
| Предусловия | Клиент зарегистрирован в ИСПУГО. |
| Постусловия | Клиент не находится в черном списке. |
| Основной поток событий | 1.0. Оказание помощи клиенту.  1. Приходит сообщение от клиента с вопросом.  2. Консультируется со специалистом  3. Дает ответ на вопрос. |
| Альтернативные  потоки событий | Альтернативного потока событий нет |
| Исключения | Клиент находится в черном списке |

Продолжение таблицы 2.6

|  |  |
| --- | --- |
| Включение | Нет |
| Приоритет | Высокий |
| Частота  использования | Приблизительно 1000 раз в день. |
| Бизнес-правила | Бизнес-правило-1. |
| Специальные  требования | Нет |
| Предположения | Клиент зарегистрирован. |
| Замечания и  вопросы | Замечаний нет. |

Таблица 2.7. Спецификация варианта использования «Просмотреть отзывы сервиса»

|  |  |
| --- | --- |
| **Название поля шаблона** | **Значение поля** |
| № прецедента | Вариант использования 1 |
| Название  прецедента | Просмотр отзывов о сервиса |
| Автор | Саевский О.В. |
| Дата создания | 02.12.2021 г. |
| Актёр | Клиент |
| Описание | Клиент запускает приложение ИСПУГО через интернет или с домашнего компьютера, выбирает сервис и просматривает отзывы о нём от других пользователей |
| Предусловия | - |
| Постусловия | 1. Пользователь не в черном списке. |
| Основной поток событий | * 1. Открытие страницы отзывов о сервисе.   1. Выбирает сервис предприятия  1.Смотрит общий рейтинг сервиса.  2.Смотрит отзывы о сервисе. |
| Альтернативные потоки событий | 1. Открытие страницы отзывов.  2. Просмотр отзывов о предприятии как большой сети. |

Продолжение таблицы 2.7

|  |  |
| --- | --- |
| Исключения | Клиент находится в черном списке |
| Включение | Нет |
| Приоритет | Средний |
| Частота  использования | Приблизительно 10000 пользователей в день. |
| Бизнес-правила | Бизнес-правило-1, Бизнес-правило-3 |
| Специальные требования | - |
| Предположения | 1.Предполагается, что 80% клиентов , которые будут в будущем заказывать услугу перед этим просмотрят отзывы |
| Замечания и вопросы | - |

Таблица 2.8. Спецификация варианта использования «Прохождение верификации»

|  |  |
| --- | --- |
| **Название поля шаблона** | **Значение поля** |
| № прецедента | Вариант использования 1 |
| Название  прецедента | Прохождение верификации |
| Автор | Саевский О.В. |
| Дата создания | 02.12.2021 г. |
| Актёр | Клиент |
| Описание | Клиент запускает приложение ИСПУГО через интернет или с домашнего компьютера, заходит в личный кабинет и проходит верификацию. |
| Предусловия | 1. Клиент авторизован. |
| Постусловия | 1. Пользователь не в черном списке. |
| Основной поток событий | 1.0 Открытие страницы личного кабинета.  1. Выбирает процесс прохождения верификации.  2. Проходит верификацию. |
| Альтернативные потоки событий | - |

Продолжение таблицы 2.8

|  |  |
| --- | --- |
| Исключения | Клиент находится в черном списке |
| Включение | Нет |
| Приоритет | Средний |
| Частота  использования | Приблизительно 300 пользователей в день. |
| Бизнес-правила | Бизнес-правило-1, Бизнес-правило-3 |
| Специальные требования | - |
| Предположения | 1.Предполагается, что 1% клиентов , которые будут в будущем заказывать услугу пройдут верификацию. |
| Замечания и вопросы | - |

Таблица 2.9. Спецификация варианта использования «Оплатить заказ»

|  |  |
| --- | --- |
| **Название поля шаблона** | **Значение поля** |
| № прецедента | Вариант использования 1 |
| Название  прецедента | Оплатить заказ |
| Автор | Саевский О.В. |
| Дата создания | 02.12.2021 г. |
| Актёр | Клиент |
| Описание | Клиент запускает приложение ИСПУГО через интернет или с домашнего компьютера, заходит в личный кабинет и оплачивает заказ. |
| Предусловия | 1. Клиент верифицирован.  2. Клиент авторизован.  3. Заказ создан. |
| Постусловия | 1. Пользователь не в черном списке. |
| Основной поток событий | 1.0 Открытие страницы личного кабинета.  1. Выбирает заказ, который нужно оплатить.  2. Оплачивает заказ. |
| Альтернативные потоки событий | - |
| Исключения | 1. У Клиента нет денег.  2. Клиент находится в черном списке |

Продолжение таблицы 2.9

|  |  |
| --- | --- |
| Включение | Нет |
| Приоритет | Средний |
| Частота  использования | Приблизительно 1000 пользователей в день. |
| Бизнес-правила | Бизнес-правило-2 |
| Специальные требования | - |
| Предположения | 1.Предполагается, что 90% клиентов , которые будут в будущем заказывать услугу её оплатят. |
| Замечания и вопросы | - |

Таблица 2.10. Спецификация варианта использования «Отменить заказ»

|  |  |
| --- | --- |
| **Название поля шаблона** | **Значение поля** |
| № прецедента | Вариант использования 1 |
| Название  прецедента | Отменить заказ |
| Автор | Саевский О.В. |
| Дата создания | 02.12.2021 г. |
| Актёр | Клиент |
| Описание | Клиент запускает приложение ИСПУГО через интернет или с домашнего компьютера, заходит в личный кабинет и отменяет заказ. |
| Предусловия | 1. Клиент авторизован.  2. Заказ существует. |
| Постусловия | - |
| Основной поток событий | 1.0 Открытие страницы личного кабинета.  1. Выбирает заказ, который нужно отменить.  2. Отменяет заказ. |
| Альтернативные потоки событий | - |
| Исключения | - |
| Включение | Нет |
| Приоритет | Средний |

Продолжение таблицы 2.10

|  |  |
| --- | --- |
| Частота  использования | Приблизительно 100 пользователей в день. |
| Бизнес-правила | Бизнес-правило-2 |
| Специальные требования | - |
| Предположения | 1.Предполагается, что 10% клиентов , которые будут в будущем заказывать услугу её отменят. |
| Замечания и вопросы | - |

Таблица 2.11. Спецификация варианта использования «Написать письмо на почту»

|  |  |
| --- | --- |
| **Название поля шаблона** | **Значение поля** |
| № прецедента | Вариант использования 1 |
| Название  прецедента | Написать письмо на почту |
| Автор | Саевский О.В. |
| Дата создания | 02.12.2021 г. |
| Актёр | Клиент |
| Описание | Клиент запускает приложение ИСПУГО через интернет или с домашнего компьютера, заходит на почтовый сервис и пишет письмо в службу поддержки предприятия. |
| Предусловия | - |
| Постусловия | Клиент не находится в черном списке |
| Основной поток событий | 1.0 Открытие страницы написания письма и отправка письма.  1. Выбирает почтовый адрес предприятия.  2. Пишет в почтовую службу поддержки предприятия. |
| Альтернативные потоки событий | - |
| Исключения | Клиент находится в черном списке |
| Включение | Нет |
| Приоритет | Низкий |
| Частота  использования | Приблизительно 100 пользователей в день. |

Продолжение таблицы 2.11

|  |  |
| --- | --- |
| Бизнес-правила | Бизнес-правило-2 |
| Специальные требования | - |
| Предположения | 1.Предполагается, что 1% клиентов будут писать письмо на почту. |
| Замечания и вопросы | - |

Таблица 2.12. Спецификация варианта использования «Просмотр списка сотрудников»

|  |  |
| --- | --- |
| **Название поля шаблона** | **Значение поля** |
| № прецедента | Вариант использования 1 |
| Название  прецедента | Просмотр списка сотрудников |
| Автор | Саевский О.В. |
| Дата создания | 02.12.2021 г. |
| Актёр | Клиент |
| Описание | Клиент запускает приложение ИСПУГО через интернет или с домашнего компьютера, выбирает сервис и просматривает список сотрудников сервиса. |
| Предусловия | - |
| Постусловия | - |
| Основной поток событий | 1.0 Выбор сервиса и просмотр списка сотрудников.  1. Выбирает сервис.  2. Просматривает список сотрудников предприятия. |
| Альтернативные потоки событий | - |
| Исключения | - |
| Включение | Нет |
| Приоритет | Низкий |
| Частота  использования | Приблизительно 1000 пользователей в день. |
| Бизнес-правила | Бизнес-правило-1 |
| Специальные требования | - |

Продолжение таблицы 2.12

|  |  |
| --- | --- |
| Предположения | 1.Предполагается, что 3% клиентов будут просматривать список сотрудников. |
| Замечания и вопросы | - |

Таблица 2.13. Спецификация варианта использования «Просмотреть услуги сервиса»

|  |  |
| --- | --- |
| **Название поля шаблона** | **Значение поля** |
| № прецедента | Вариант использования 1 |
| Название  прецедента | Просмотреть услуги сервиса |
| Автор | Саевский О.В. |
| Дата создания | 02.12.2021 г. |
| Актёр | Клиент |
| Описание | Клиент запускает приложение ИСПУГО через интернет или с домашнего компьютера, выбирает сервис предприятия и просматривает его услуги. |
| Предусловия | - |
| Постусловия | - |
| Основной поток событий | 1.0 Выбор сервиса предприятия и просмотр услуг.  1. Выбор сервиса.  2. Просмотр услуг сервиса. |
| Альтернативные потоки событий | - |
| Исключения | - |
| Включение | Нет |
| Приоритет | Высокий |
| Частота  использования | Приблизительно 100000 пользователей в день. |
| Бизнес-правила | Бизнес-правило-2 |
| Специальные требования | - |
| Предположения | 1.Предполагается, что 10% клиентов будут звонить менеджерам. |

Продолжение таблицы 2.13

|  |  |
| --- | --- |
| Замечания и вопросы | - |

Таблица 2.14. Спецификация варианта использования «Позвонить менеджеру»

|  |  |
| --- | --- |
| **Название поля шаблона** | **Значение поля** |
| № прецедента | Вариант использования 1 |
| Название  прецедента | Позвонить менеджеру |
| Автор | Саевский О.В. |
| Дата создания | 02.12.2021 г. |
| Актёр | Клиент |
| Описание | Клиент запускает приложение ИСПУГО через интернет или с домашнего компьютера, звонит менеджеру и задает вопрос. |
| Предусловия | - |
| Постусловия | - |
| Основной поток событий | 1.0 Звонок менеджера с целью задать вопрос.  1. Звонит менеджеру.  2. Задает вопрос. |
| Альтернативные потоки событий | - |
| Исключения | - |
| Включение | Нет |
| Приоритет | Низкий |
| Частота  использования | Приблизительно 5000 пользователей в день. |
| Бизнес-правила | Бизнес-правило-2 |
| Специальные требования | - |
| Предположения | 1.Предполагается, что 10% клиентов будут звонить менеджерам. |
| Замечания и вопросы | - |

## 2.5 Тип приложения

В ходе проектирования проекта было выбрано веб приложение.

Приложения будет доступно по адресу в сети, который установит предприятие на сервере. Это гарантирует кроссплатформенность и корректное отображение на всех устройствах, у которых есть доступ в Интернет.

## 2.6 Определение ограничений развёртывания

При проектировании были выделены следующие ограничения развёртывания:

*Ограничения по аппаратным требованиям*: наличие процессора с тактовой частотой 1.3 Гц.

*Ограничения по программным требованиям*: Android 4.0 или выше, IOS, Windows XP или выше, Linux, MacOS.

*Ограничения собственно по развёртыванию*. Использования нескольких файловых серверов, наличие веб-сервера, сервера с БД, использование ОС Linux, сервер для резервного копирования, веб-сервер для прокси.

*Повышенные требования безопасности*. Применение технологии SSH, обеспечивающего https протокол.

## 2.7 Архитектурные стили проектирования

В ходе проектирования системы были использованы следующие стили проектирования:

*- Репозиторий*. Все совместно используемые подсистемами данные хранятся в центральной БД, API которой реализуется с помощью репозитория.

*- Паттерн Цепочка-обязанностей*. При запросе на сервер прокси или главный сервер ресурса до адресации к нужному контроллеру сервера необходимо проверить запрос на содержание. Это поможет обработать запрос на авторизацию или другой контекст еще до основной логики работы контроллера. Тем самым мы уменьшим количество запросов к БД, следовательно и нагрузку на сервера.

## 2.8 Графическое представление архитектуры

Структура (высокоуровневая архитектура) декомпозиции ПС представлена на рисунке 2.3.

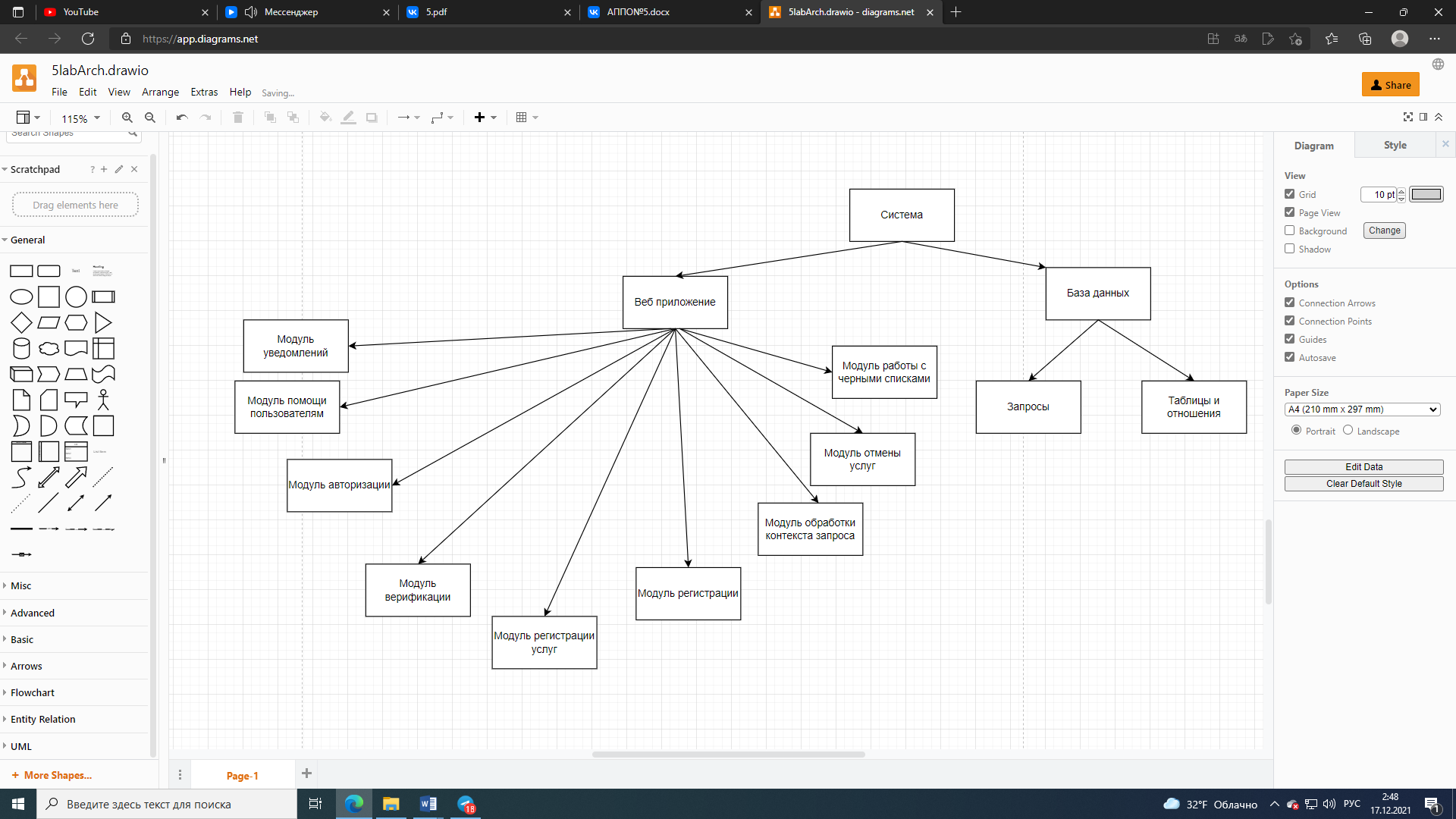


Рисунок 2.3 – Структура декомпозиции ПС

Отображение полученного варианта архитектуры в других представлениях (модульном, классах ООП, структурах «компонент и соединитель», структурах «распределение») с использование диаграмм Use Case представлены на рисунках 2.4 – 2.8.

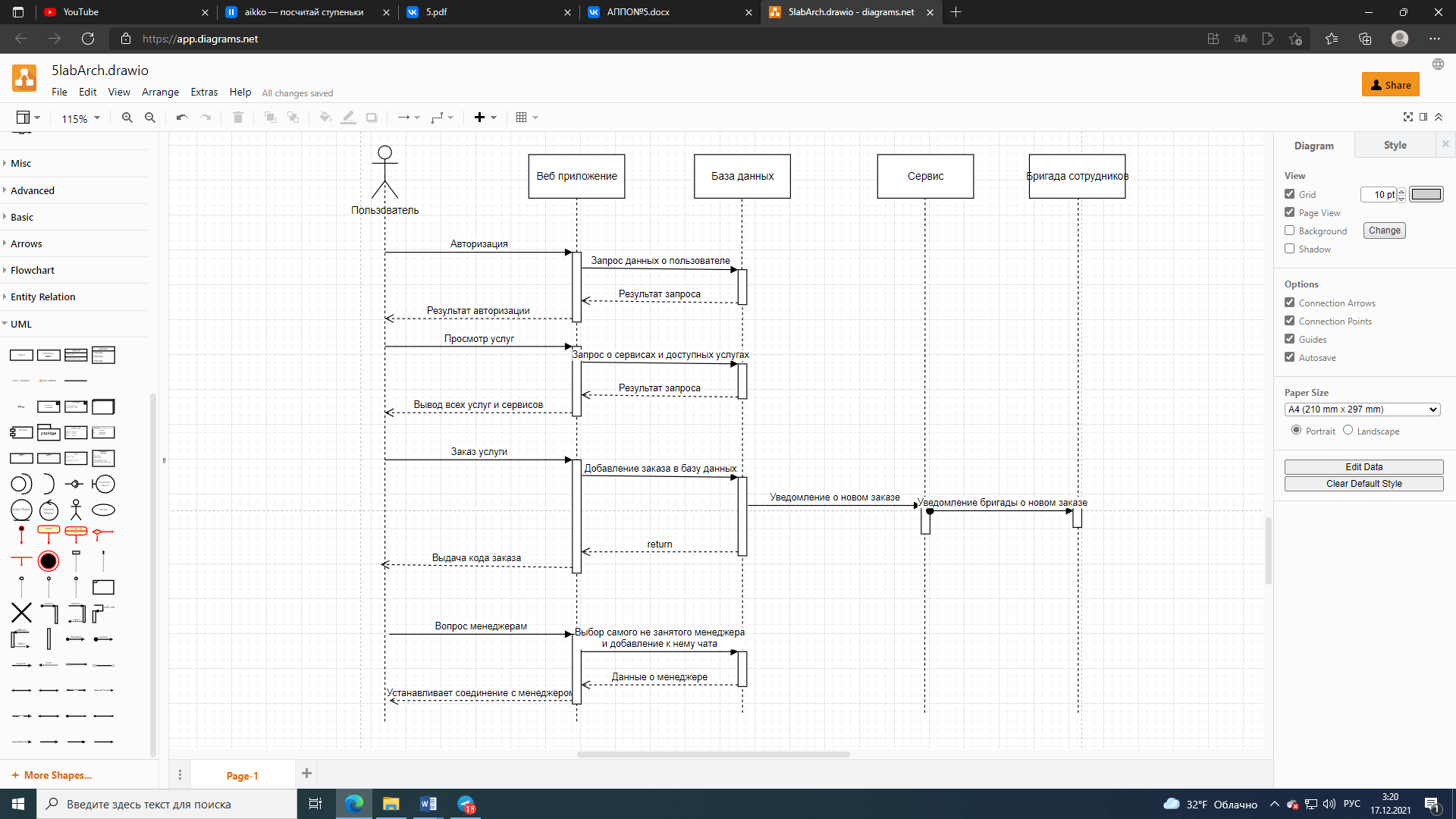


Рисунок 2.4 – Отображение полученного варианта архитектуры в виде диаграммы последовательности

Формула и расчёты для диаграммы последовательности.

Диаграмма вошла в диапазон между 3 и 3.5, оценка допустима.

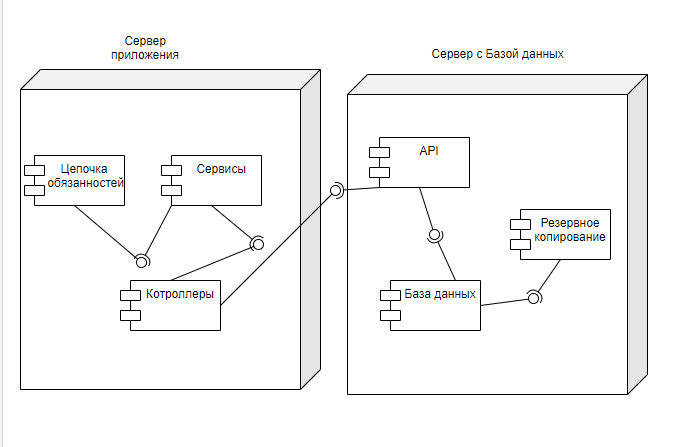


Рисунок 2.5 – Отображение полученного варианта архитектуры в виде диаграммы развертывания

Формула и расчёты для диаграммы развёртывания.

Диаграмма вошла в диапазон между 2 и 2.5, оценка допустима.



Рисунок 2.6 – Отображение полученного варианта архитектуры в виде диаграммы прецедентов

Формула и расчёты для диаграммы прецедентов.

Диаграмма вошла в диапазон между 2.5 и 3, оценка допустима.

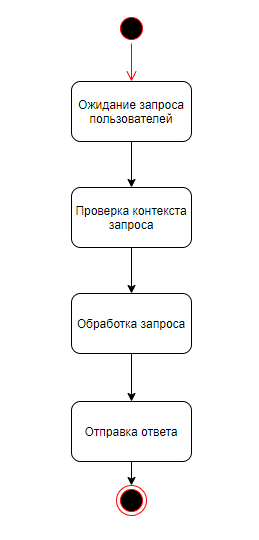


Рисунок 2.7 – Диаграмма состояний системы

Формула и расчёты для диаграммы состояний.

Диаграмма вошла в диапазон между 2.5 и 3, оценка допустима.

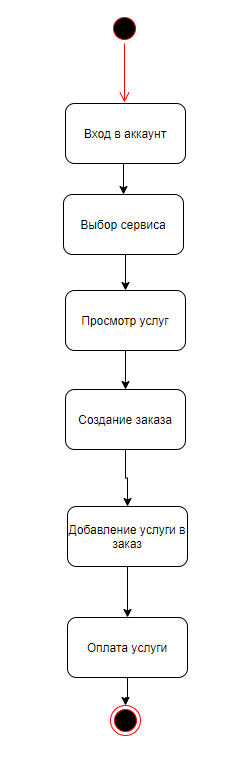


Рисунок 2.8 – Диаграмма активностей

Формула и расчёты для диаграммы активностей.

Диаграмма вошла в диапазон между 2.5 и 3, оценка допустима.

## 2.9 Детализация программ с помощью паттернов проектирования

Описание паттернов и детализация программ с помощью паттернов представлено в таблицах 2.15-2.16.

Таблица 2.15 – Описание паттерна репозиторий

|  |  |
| --- | --- |
| Описание | Все совместно используемые подсистемами данные хранятся в центральной базе данных, доступной всем подсистемам. Репозиторий является пассивным элементом, а управление им возложено на подсистемы. |
| Преимущества | Совместное использование больших объёмов данных эффективно, поскольку не требуется передавать данные из одной подсистемы в другие. Подсистема не должна знать, как используются данные в других подсистемах - уменьшается степень связывания.  В системах с репозиторием резервное копирование, обеспечение безопасности, управление доступом и восстановление данных централизованы, поскольку входят в систему управления репозиторием. |
| Недостатки | Все подсистемы должны быть согласованы со структурой репозитория (моделью данных). Модернизировать модель данных достаточно трудно  К разными подсистемам предъявляются различные требования по безопасности, восстановлению и резервированию данных, а в паттерне Репозиторий ко всем подсистемам применяется одинаковая политика. |

Таблица 2.16 – Описание паттерна цепочка обязанностей

|  |  |
| --- | --- |
| Описание | Паттерн позволяет избежать жёсткой зависимости отправителя запроса от его получателя, при этом запрос может быть обработан несколькими объектами. Объекты-получатели связываются в цепочку. Запрос передаётся по этой цепочке, пока не будет обработан. Он вводит конвейерную обработку для запроса с множеством возможных обработчиков. Цепочка – это объектно-ориентированный связанный список с рекурсивным обходом. |
| Преимущества | Избегание жёсткой зависимости отправителя запроса от его получателя дает доступ к реализации полиморфизма, что в свою очередь является преимуществом для расширяемости системы. |
| Недостатки | Каждое звено цепочки обрабатывает единый контекст запроса и может его изменять на своё усмотрение. Таким образом контекст запроса – глобальная переменная, которая в случае неверного изменения может сломать работу программы. |

## 2.10 Пользовательский интерфейс

Пользовательский интерфейс представлен на рисунке 2.9.

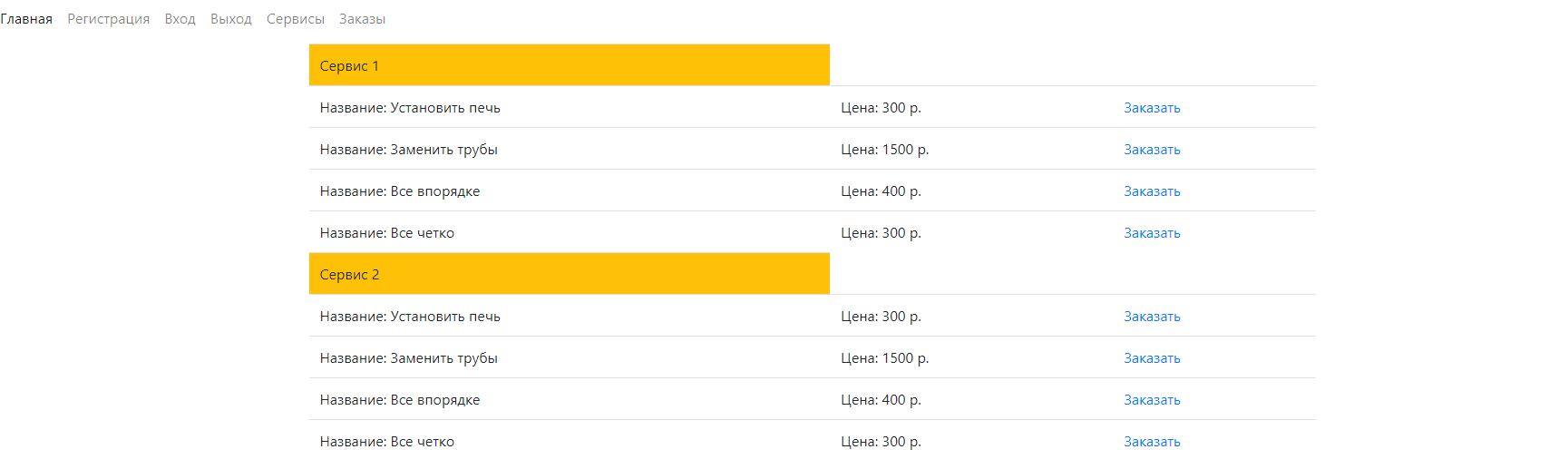


Рисунок 2.9 – Прототип интерфейса

## 2.11 Инструментальные средства разработки ПС.

Интегрированная среда разработки (IDE) — это многофункциональная программа, которая поддерживает многие аспекты разработки программного обеспечения. Интегрированная среда разработки Visual Studio — это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Помимо стандартного редактора и отладчика, которые есть в большинстве сред IDE, Visual Studio включает в себя компиляторы, средства автодополнения кода, графические конструкторы и многие другие функции для улучшения процесса разработки.

Draw.io — это сервис, предназначенный для формирования диаграмм и схем. Сервис разделён на три части — меню, панель объектов и сам документ.

С помощью веб-сервиса Draw.io можно создавать:

* диаграммы;
* UML-модели;
* вставка в диаграмму изображений;
* графики;
* блок-схемы;
* формы;

PostgreSQL — свободная объектно-реляционная система управления базами данных (СУБД). Существует в реализациях для множества UNIX-подобных платформ, включая AIX, различные BSD-системы, HP-UX, IRIX, Linux, macOS, Solaris/OpenSolaris, Tru64, QNX, а также для Microsoft Windows.

## 2.12 Программная реализация базовых модулей системы

В ходе реализации системы были созданы следующие классы: Service.cs, Order.cs, ServiceTask, PostgresConnection.cs, OrderRepository.cs, ServiceTaskRepository.cs, ServiceRepository.cs, AbstractMiddleware.cs, CheckServiceDataMiddleware.cs, CheckServiceInDbMiddleware.cs, Program.cs. Описание данных классов представлено в таблице 2.17.

Таблица 2.17 – Описание классов системы

|  |  |
| --- | --- |
| Класс | Описание |
| Service.cs | Является моделью представления сервиса предприятия. Методов не имеет. |
| Order.cs | Является моделью представления заказа предприятия. Методов не имеет |
| PostgresConnection.cs | Является моделью представления базы данных. |
| ServiceRepository | Является классом для работы с моделью Service. Содержит методы добавления и загрузки объектов |
| Program.cs | Собирает веб приложение, пробрасывает порты, собирает контейнер зависимостей. |
| AbstractMiddleware.cs | Абстрактный класс реализации звена обязанностей |
| CheckTaskDataMiddleware.cs | Звено обязанности, которое проверяет корректность данных объекта услуги |
| CheckTaskInDbMiddleware.cs | Звено обязанности, которое проверяет наличие модели в базе данных |

Продолжение таблицы 2.17

|  |  |
| --- | --- |
| ServiceTask.cs | Является моделью представления услуги предприятия. Методов не имеет. |
| Order.cs | Является моделью представления заказа предприятия. Методов не имеет |
| OrderRepository.cs | Является классом для работы с моделью Order. Содержит методы добавления и загрузки объектов |
| ServiceTaskRepository.cs | Является классом для работы с моделью ServiceTask. Содержит методы добавления и загрузки объектов |

Описание функций системы представлено в таблице 2.18.

Таблица 2.18 – Описание функций системы

|  |  |  |  |
| --- | --- | --- | --- |
| Вызов | Параметры | Результат | Описание |
| Create | object | - | Создает объект модели |
| GetAll | - | IEnumerable<T> | Возвращает список всех объектов типа T из базы данных |
| Get | id | T object | Возвращает объект типа T из БД с указанным id |
| SaveAsync | - | - | Сохраняет конфигурацию базы данных |
| DisposeAsync | - | - | Высвобождает ресурсы БД из памяти |
| Delete | object | - | Асинхронно удаляет объект из базы данных |
| ReadAll | - | Список услуг | Загружает из базы данных все услуги |
| Save | - | - | Сохраняет изменения в БД |
| Execute | - | - | Выполняет алгоритм звена обязанности |

Диаграмма классов представлена на рисунке 2.10.

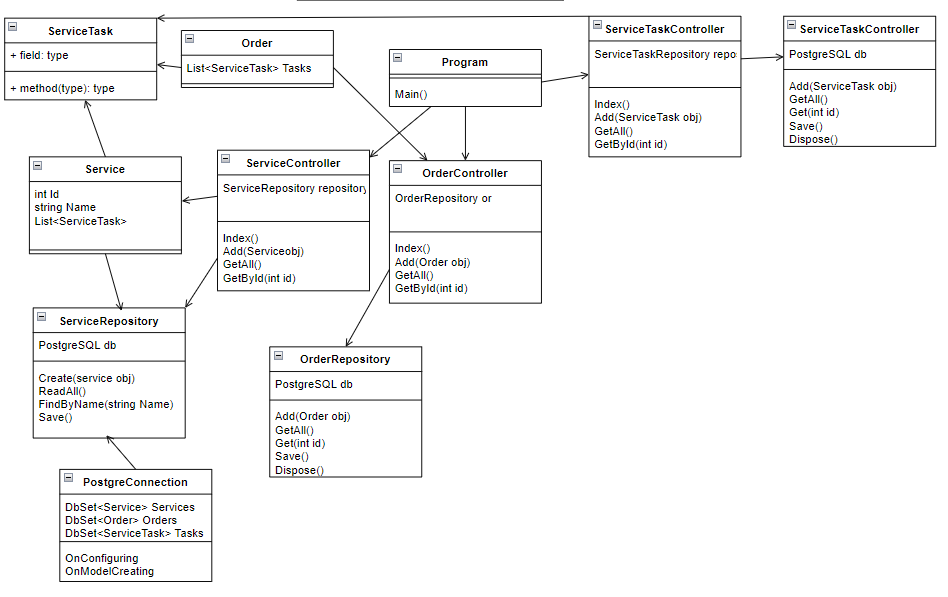


Рисунок 2.10 – Диаграмма классов

Формула и расчёты для диаграммы классов.

Диаграмма вошла в диапазон между 5 и 5.5, оценка допустима.

## 2.13 Анализ качества и оценка программного дизайна

*Параметры качества* - это общие свойства архитектуры, которые оказывают влияние на дизайн системы, ее поведение во время работы и взаимодействие с пользователем, такие как: удобство и простота использования, производительность, надежность и безопасность и др. Обеспечение приложением требуемого сочетания параметров качества определяет успешность его дизайна и общее качество программного продукта. В ходе проектирования приложения, отвечающего любому из этих параметров, необходимо учесть влияние и других требований, а также при этом должны быть проанализированы плюсы и минусы по отношению к другим параметрам качества.

*Атрибуты качества дизайна.* Существует целый спектр различных атрибутов, помогающих оценить разработку и добиться качественного дизайна. Эти атрибуты могут описывать многие характеристики системы и элементов дизайна: «тестируемость», «переносимость», «модифицируемость», «производительность», «безопасность» и т. п. Важно понимать, что обсуждаемые атрибуты касаются только дизайна (как результата), но не проектирования (как процесса). Все эти атрибуты принято объединять в следующие группы:

- применимые к run-time, т. е. ко времени выполнения системы, например, среднее время отклика системы, позволяющее оценить качество дизайна с точки зрения производительности;

- ориентированные на design-time, т. е. позволяющие оценивать качество получаемого дизайна еще на этапе проектирования и в общем случае, вплоть до тестирования включительно; например, средняя загруженность классов бизнес-методами (в каждом классе бизнес-методов «в среднем» или «много»);

- атрибуты качества архитектурного дизайна как такового: концептуальная целостность дизайна, непротиворечивость, полнота, завершенность; например, любой определенный бизнес-метод является вызываемым, т. е. создан не потому, что может понадобиться в будущем, а определен в соответствии с требованиями или необходим для реализации дизайна в выбранном архитектурном стиле.

Существуют атрибуты, которые сложно измерить, например, порти- руемость (переносимость на другие операционные системы и т. д.) или безопасность. Измеряемые атрибуты качества описываются определенными метриками. Метрика позволяет количественно оценить атрибут качества, например, «модифицируемость» и «сложность» системы.

Не стоит путать атрибуты качества дизайна с атрибутами качества, используемыми в ряде требований, предъявляемых к системе. Часть из них может отображаться друг на друга и нести эквивалентную смысловую нагрузку, некоторые могут быть связаны, однако большая часть атрибутов качества дизайна является специфичной именно для дизайна и не связана с требованиями. Например, если используется платформа J2EE (Java 2 Enterprise Edition) и компонентная модель EJB (Enterprise JAVABEANS), существуют признаки хорошего дизайна, специфичные для данной платформы и компонентной модели, но абсолютно никак не связанные с какими-либо требованиями к создаваемой на этой платформе программной системе.

*Анализ качества и техники оценки.* В индустрии разработки ПО распространены многие инструменты, техники и практики, помогающие добиться качественного дизайна:

- обзор дизайна (software design review), например, неформальный обзор архитектуры членами проектной команды;

- статический анализ (static analysis), например трассировка с требованиями

- симуляция и прототипирование (simulation and prototyping) - динамические техники проверки дизайна в целом или отдельных его атрибутов качества.

## 2.14 Анализ проектирования

Проектирование – процесс, при котором происходит подбор всех моделей, архитектуры и характеристик будущего проекта.

Проектирование программного обеспечения — процесс создания проекта программного обеспечения (ПО), а также дисциплина, изучающая [методы проектирования](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D1%8B_%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F). Проектирование ПО является частным случаем [проектирования](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) продуктов и процессов.

Анализ проектирования системы представлен в таблицах 2.18 – 2.23.

Таблица 2.18 - Оценочные элементы фактора "Надежность ПС"

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| Н0101 | Наличие требований к программе по устойчивости функционирования при наличии ошибок во входных данных | Экспертный | 1 |
| Н0102 | Возможность обработки ошибочных ситуаций | То же | 1 |
| Н0103 | Полнота обработки ошибочных ситуаций | " | 0,8 |
| Н0104 | Наличие тестов для проверки допустимых значений входных данных |  | 1 |
| Н0105 | Наличие системы контроля полноты входных данных |  | 1 |
| Н0106 | Наличие средств контроля корректности входных данных |  | 1 |
| Н0107 | Наличие средств контроля непротиворечивости входных данных |  | 1 |
| Н0109 | Наличие обработки граничных результатов |  | 1 |
| Н0110 | Наличие обработки неопределенностей (указание одного гражданства с предъявлением паспорта другого). |  | 0.8 |

Продолжение таблицы 2.18

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Н0201 | Наличие требований к программе по восстановлению процесса выполнения в случае сбоя операционной системы, процессора, внешних устройств |  | 1 | |
| H0202 | Наличие требований к программе по восстановлению результатов при отказах процессора, ОС |  | 1 | |
| Н0203 | Наличие средств восстановления процесса в случае сбоев оборудования |  | 0,4 | |
| Н0204 | Наличие возможности разделения по времени выполнения отдельных функций программ |  | 1 |
| Н0205 | Наличие возможности повторного старта с точки останова |  | 0.5 |
| H0301 | Наличие централизованного управления процессами, конкурирующими из-за ресурсов |  | 0.7 |

Продолжение таблицы 2.18

|  |  |  |  |
| --- | --- | --- | --- |
| H0302 | Наличие возможности автоматически обходить ошибочные ситуации в процессе вычисления |  | 1 |
| Н0303 | Наличие средств, обеспечивающих завершение процесса решения в случае помех |  | 1 |
| H0304 | Наличие средств, обеспечивающих выполнение программы в сокращенном объеме в случае ошибок или помех |  | 0.8 |
| Н0305 | Показатель устойчивости к искажающим воздействиям | Расчетный | https://api.docs.cntd.ru/img/12/00/00/91/35/34ee1e2c-6c63-4430-8809-58fff8c40c7a/P011300530000.png, P(Y) = 1 – 47/50 = 0,96 |

Таблица 2.19 - Оценочные элементы фактора "Сопровождаемость"

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| C0101 | Наличие модульной схемы программы |  | 1 |
| C0102 | Оценка программы по числу уникальных модулей |  | 1 |
| C0201 | Наличие ограничений на размеры модуля |  | 1 |

Продолжение таблицы 2.19

|  |  |  |  |
| --- | --- | --- | --- |
| C0301 | Наличие проверки корректности передаваемых данных |  | 1 |
| С0302 | Оценка простоты программы по числу точек входа и выхода | Расчетный | https://api.docs.cntd.ru/img/12/00/00/91/35/34ee1e2c-6c63-4430-8809-58fff8c40c7a/P0117000B0000.png, = |
| С0303 | Осуществляется ли передача результатов работы модуля через вызывающий его модуль | Экспертный | 1 |
| С0304 | Осуществляется ли контроль за правильностью данных, поступающих в вызывающий модуль от вызываемого |  | 0,7 |
| С030 | Наличие требований к независимости модулей программы от типов и форматов выходных данных |  | 1 |

Таблица 2.20 - Оценочные элементы фактора "Удобство применения"

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| У0801 | Уровень языка общения пользователя с программой |  | 1 |

Продолжение таблицы 2.20

|  |  |  |  |
| --- | --- | --- | --- |
| У0802 | Легкость и быстрота загрузки и запуска программы |  | 0,9 |
| У0803 | Легкость и быстрота завершения работы программы |  | 0,2 |
| У0804 | Возможность распечатки содержимого чека заказа |  | 1 |
| У0805 | Возможность приостанова и повторного запуска работы без потерь информации |  | 1 |
| У0901 | Соответствие меню требованиям пользователя |  | 0.6 |
| У0902 | Возможность прямого перехода вверх и вниз по многоуровнему меню (пропуск уровней) |  | 1 |
| У1001 | Возможность управления подробностью получаемых выходных данных |  | 0.8 |
| У1002 | Достаточность полученной информации для продолжения работы |  | 1 |

Продолжение таблицы 2.20

|  |  |  |  |
| --- | --- | --- | --- |
| У1101 | Обеспечение удобства ввода данных |  | 1 |
| У1102 | Легкость восприятия |  | 1 |
| У1201 | Обеспечение программой выполнения предусмотренных рабочих процедур |  | 1 |
| У1202 | Достаточность информации, выдаваемой программой для составления дополнительных процедур |  | 1 |

Таблица 2.21 - Оценочные элементы фактора "Эффективность"

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| Э0101 | Проблемно-ориентированные функции | Экспертный или расчетный | 1 |
| Э0102 | Машинно-ориентированные функции | То же | 0,6 |
| Э0103 | Функции ведения и управления |  | 1 |
| Э0104 | Функции ввода/вывода |  | 1 |
| Э0105 | Функции защиты и проверки данных |  | 1 |
| Э0106 | Функции защиты от несанкционированного доступа |  | 1 |
| Э0107 | Функции контроля доступа |  | 1 |
| Э0108 | Функции защиты от внесения изменений |  | 1 |

Продолжение таблицы 2.21

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Э0109 | Наличие соответствующих границ функциональных областей | | |  | | 1 |
| Э0110 | Число знаков после запятой в результатах вычислений | | |  | | 0,9 |
| Э0201 | Время выполнения программ | | |  | | 0,7 |
| Э0202 | Время реакции и ответов | | |  | | 1 |
| Э0203 | Время подготовки | | |  | | 1 |
| Э0205 | Затраты времени на защиту данных | | |  | | 1 |
| Э0206 | | Время компиляции |  | | 0.6 | |
| Э0301 | | Требуемый объем внутренней памяти |  | | 0.7 | |
| Э0302 | | Требуемый объем внешней памяти |  | | 1 | |
| Э0303 | | Требуемые периферийные устройства |  | | 1 | |
| Э0304 | | Требуемое базовое программное обеспечение |  | | 1 | |

Таблица 2.22 - Оценочные элементы фактора "Универсальность"

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| Г0101 | Оценка числа потенциальных пользователей | Экспертный | 1 |
| Г0102 | Оценка числа функций ПС | То же | 1 |
| Г0103 | Насколько набор функций удовлетворяет требованиям пользователя |  | 1 |

Продолжение таблицы 2.22

|  |  |  |  |
| --- | --- | --- | --- |
| Г0104 | Насколько возможности программ охватывают область решаемых пользователем задач |  | 1 |
| Г0105 | Возможность настройки формата выходных данных для конкретных пользователей |  | 1 |
| Г0201 | Наличие схемы иерархии модулей программы |  | 1 |
| Г0202 | Оценка независимости модулей |  | 0,8 |
| Г0203 | Оценка числа уникальных элементов/реквизитов |  | 1 |
| Г0204 | Используется ли в текущем вызове модуля информация, полученная в предыдущем вызове |  | 0.5 |
| Г0205 | Оценка организации точек входа и выхода модуля |  | 1 |
| Г0206 | Наличие описания атрибутов модуля |  | 1 |
| Г0301 | Оценка программ по числу переходов и точек ветвления |  | 1 |
| Г0701 | Оценка зависимости программ от емкости оперативной памяти ЭВМ |  | 0.6 |

Продолжение таблицы 2.22

|  |  |  |  |
| --- | --- | --- | --- |
| Г0702 | Оценка зависимости временных характеристик программы от скорости вычислений ЭВМ |  | 1 |
| Г0703 | Оценка зависимости функционирования программы от числа внешних запоминающих устройств и их общей емкости |  | 1 |
| Г0704 | Оценка зависимости функционирования программы от специальных устройств ввода-вывода |  | 0.3 |
| Г0801 | Применение специальных языков программирования |  | 1 |
| Г0802 | Оценка зависимости программы от программ операционной системы |  | 0.5 |
| Г0803 | Зависимость от других программных средств |  | 0.5 |
| Г0901 | Оценка локализации непереносимой части программы |  | 1 |

Таблица 2.23 - Оценочные элементы фактора "Корректность"

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| К0101 | Наличие всех необходимых документов для понимания и использования ПС | Экспертный | 1 |

Продолжение таблицы 2.23

|  |  |  |  |
| --- | --- | --- | --- |
| К0102 | Наличие описания и схемы иерархии модулей программы |  | 1 |
| К0103 | Наличие описания основных функций |  | 1 |
| К0104 | Наличие описания частных функций |  | 1 |
| К0105 | Наличие описания данных |  | 1 |
| К0106 | Наличие описания алгоритмов |  | 1 |
| К0107 | Наличие описания интерфейсов между модулями |  | 1 |
| К0108 | Наличие описания интерфейсов с пользователями |  | 1 |
| К0109 | Наличие описания используемых числовых методов |  | 1 |
| К0110 | Указаны ли все численные методы |  | 1 |
| К0111 | Наличие описания всех параметров |  | 1 |
| К0112 | Наличие описания методов настройки системы |  | 1 |
| К0113 | Наличие описания всех диагностических сообщений |  | 1 |
| К0114 | Наличие описания способов проверки работоспособности программы |  | 1 |
| К0301 | Отсутствие противоречий в описании частных функций |  | 1 |

Продолжение таблицы 2.23

|  |  |  |  |
| --- | --- | --- | --- |
| К0302 | Отсутствие противоречий в описании основных функций в разных документах |  | 1 |
| К0303 | Отсутствие противоречий в описании алгоритмов |  | 1 |
| К0304 | Отсутствие противоречий в описании взаимосвязей в системе |  | 1 |
| К0305 | Отсутствие противоречий в описании интерфейсов между модулями |  | 1 |
| К0306 | Отсутствие противоречий в описании интерфейсов с пользователем |  | 1 |
| К0307 | Отсутствие противоречий в описании настройки системы |  | 1 |
| К0309 | Отсутствие противоречий в описании иерархической структуры сообщений |  | 1 |
| К0310 | Отсутствие противоречий в описании диагностических сообщений |  | 1 |

Продолжение таблицы 2.23

|  |  |  |  |
| --- | --- | --- | --- |
| К0311 | Отсутствие противоречий в описании данных |  | 1 |
| К0501 | Единообразие способов вызова модулей |  | 1 |
| К0502 | Единообразие процедур возврата управления из модулей |  | 1 |
| К0503 | Единообразие способов сохранения информации для возврата |  | 1 |
| К0504 | Единообразие способов восстановления информации для возврата |  | 1 |
| К0505 | Единообразие организации списков передаваемых параметров |  | 1 |
| К0601 | Единообразие наименования каждой переменной и константы |  | 1 |
| К0602 | Все ли одинаковые константы встречаются во всех программах по одинаковыми именами |  | 1 |
| К0603 | Единообразие определения внешних данных во всех программах |  | 1 |

Продолжение таблицы 2.23

|  |  |  |  |
| --- | --- | --- | --- |
| К0604 | Используются ли разные идентификаторы для разных переменных |  | 1 |
| К0605 | Все ли общие переменные объявлены как общие переменные |  | 1 |
| К0606 | Наличие определений одинаковых атрибутов |  | 1 |
| К0701 | Комплектность документации в соответствии со стандартами |  | 1 |
| К0702 | Правильное оформление частей документов |  | 1 |
| К0703 | Правильное оформление титульных и заглавных листов документов |  | 1 |
| К0704 | Наличие в документах всех разделов соответствии со стандартами |  | 1 |
| К0705 | Полнота содержания разделов в соответствии со стандартами |  | 1 |

Продолжение таблицы 2.23

|  |  |  |  |
| --- | --- | --- | --- |
| К0706 | Деление документов на структурные элементы: разделы, подразделы, пункты, подпункты |  | 1 |
| К0801 | Соответствии организации и вычислительного процесса эксплуатационной документации |  | 1 |
| К0802 | Правильность заданий на выполнение программы, правильность написания управляющие и операторов (отсутствие ошибок) |  | 1 |
| К0803 | Отсутствие ошибок в описании действий пользователя |  | 1 |
| К0804 | Отсутствие ошибок в описании запуска |  | 1 |
| К0805 | Отсутствие ошибок в описании генерации |  | 1 |
| К0806 | Отсутствие ошибок в описании настройки |  | 1 |

Продолжение таблицы 2.23

|  |  |  |  |
| --- | --- | --- | --- |
| К1001 | Наличие требований к тестированию программ |  | 1 |
| К1002 | Достаточность требований к тестированию программ |  | 1 |
| К1003 | Отношение числа модулей, отработавших в процессе тестирования и отладки () к общему числу модулей () | Расчетный | 1 |
| К1004 | Отношение числа логических блоков, отработавших в процессе тестирования и отладки (), к общему числу логических блоков в программе () | То же | 1 |

Расчёт фактора «Надёжность»

*Q=10*

*Q=5*

*=0,7*

*=0,3*

Оценка фактора «Корректность»

Критерий «Полнота реализации»

*Q=14*

P1=М1=1.

Критерий «Согласованность»

Определение

*Q=5*

Определение

*Q=6*

Определение

*Q=6*

*=0.35*

*=0.35*

*=0.3*

Критерий «Проверенность»

*Q=2*

*=0.2*

*=0.38*

*=0.42*

=0.42\*1.1+0.38\*1.1+0.2\*1.1= 1.1

Вывод: разработчик ПС обеспечил установленный уровень качества выполнения этапа ЖЦ «Анализ требований» по надёжности () и по корректности ().

# 3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ МОДУЛЯ ПРОЕКТА

## 3.1 Описание используемых паттернов

Во время разработки системы регулярно использовался структурный паттерны Репозиторий и Цепочка обязанностей.

Для каждой модели данных должен использоваться свой репозиторий, поскольку он позволяет абстрагироваться от конкретных подключений к источникам данных, с которыми работает программа, и является промежуточным звеном между классами, непосредственно взаимодействующими с данными, и остальной программой.

Допустим, у нас есть одно подключение к базе данных MS SQL Server. Однако, что если в какой-то момент времени мы захотим сменить подключение с MS SQL на другое - например, к бд MySQL или MongoDB. При стандартном подходе даже в небольшом приложении, осуществляющем выборку, добавление, изменение и удаление данных, нам бы пришлось сделать большое количество изменений. Либо в процессе работы программы в зависимости от разных условий мы хотим использовать два разных подключения. Таким образом, репозиторий добавляет программе гибкость при работе с разными типами подключений.

В текущем проекте репозиторий использовался в процессе подключения к PostgreSQL. Благодаря этому, в дальнейшем можно будет легко добавлять новые.

Помимо этого, использовался паттерн Цепочка-обязанностей. Данный паттерн применяется, когда:

- Когда имеется более одного объекта, который может обработать определенный запрос.

- Когда надо передать запрос на выполнение одному из нескольких объект, точно не определяя, какому именно объекту.

- Когда набор объектов задается динамически.

## 3.2 Структура используемых паттернов

Структура используемых паттернов представлена на рисунках 3.1-3.2.

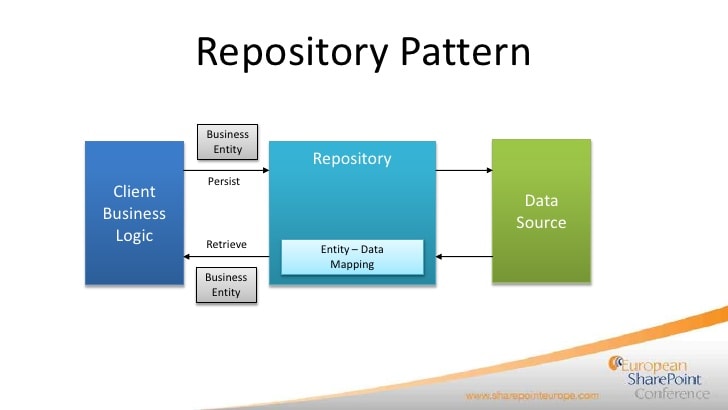


Рисунок 3.1 – Диаграмма структуры паттерна Репозиторий

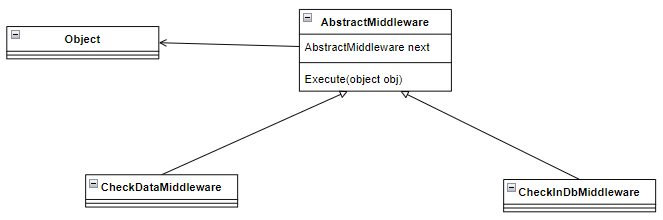


Рисунок 3.2 – Диаграмма структуры паттерна Цепочка-обязанностей

## 3.3 Программный код паттернов

namespace Website.Middleware

{

public abstract class AbstractMiddleware <T> where T : class

{

private AbstractMiddleware<T> next;

public AbstractMiddleware(AbstractMiddleware<T> next)

{

this.next = next;

}

public AbstractMiddleware() { }

protected AbstractMiddleware<T> Next => next;

public abstract Task Execute(T obj);

}

}

using Common.Models;

namespace Website.Middleware

{

public class CheackDataTaskMiddleware : AbstractMiddleware<ServiceTask>

{

public CheackDataTaskMiddleware(AbstractMiddleware<ServiceTask> next) : base(next) { }

public async override Task Execute(ServiceTask obj)

{

if (obj.Price < 0)

throw new Exception("Цена ниже нуля");

if(obj.Name == string.Empty || obj.Name == null)

throw new Exception("Название не указано");

await Next?.Execute(obj);

}

}

}

using Common.Models;

using Newtonsoft.Json;

namespace Website.Middleware

{

public class CheckTaskInDatabaseMiddleware : AbstractMiddleware<ServiceTask>

{

private HttpClient \_httpClient = new();

public override async Task Execute(ServiceTask obj)

{

var response = await \_httpClient.GetAsync($"http://localhost:7270/api/task/name={obj.Name}");

if (response.IsSuccessStatusCode)

{

var data = await response.Content.ReadAsStringAsync();

var task = JsonConvert.DeserializeObject<ServiceTask>(data);

if (task == null)

{

Next?.Execute(obj);

}

else throw new Exception("Услуга уже была создана");

}

}

}

}

using Common.Models;

using DatabaseServer.Database;

namespace DatabaseServer.Repositories

{

public class CityRepository : IRepository<City>

{

private readonly PostgresConnection \_db;

public CityRepository(PostgresConnection db)

{

\_db = db;

}

public async Task Add(City obj)

{

await \_db.Cities.AddAsync(obj);

}

public Task Delete(City obj)

{

\_db.Cities.Remove(obj);

return Task.CompletedTask;

}

public async Task Dispose()

{

await \_db.DisposeAsync();

}

public City? Get(int id)

{

return \_db.Cities.SingleOrDefault(x=>x.Id == id);

}

public IEnumerable<City> GetAll()

{

return \_db.Cities.AsEnumerable();

}

public async Task SaveAsync()

{

await \_db.SaveChangesAsync();

}

}

}

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы были закреплены практические навыки проектирования программных систем с использованием различных моделей архитектурных решений.

В результате выполнения курсовой работы были разработаны спецификации для компонентов программного продукта, были использованы различные технологи разработки программного обеспечения. Использовались формальные методы проектирования и конструирования программного обеспечения

Результатом курсового проекта является спроектированное программное обеспечение на основе паттернов.

# СПИСОК ЛИТЕРАТУРЫ

1. Крачтен Ф., Оббинк Х., Стаффорд Д. Ретроспектива программных архитектур на сайте <http://www.osp.ru> .
2. Орлов С.А., Цилькер Б.Я. Технологии разработки программного обеспечения: Учебник для вузов. 4-е изд. Стандарт третьего поколения. – СПб.: Питер, 2012. – 608 с.
3. Брауде Э. Технология разработки программного обеспечения: пер. с англ. / Э. Брауде. СПб.: Питер, 2004. - 655 с.
4. Макконнелл C. Совершенный код. Мастер-класс: пер. с англ. / С. Макконнелл. СПб.: Питер, 2007. - 896 с.
5. Макконнелл C. Сколько стоит программный проект: пер. с англ. / С. Макконнелл. СПб.: Питер, 2007. 304 с.

# 

# ПРИЛОЖЕНИЕ А

Техническое задание

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ДНР

ГОУВПО «ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра программной инженерии им. Л.П. Фельдмана

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ: «Создание информационной системы предприятия по установке и обслуживанию газового оборудования»

ПО КУРСУ: «Архитектура и проектирования программного обеспечения»

Выдано:

студенту группы ПИ-19а

Саевскому О.В.

Руководители:

Серёженко О.А.

Ищенко А.П.

ДОНЕЦК – 2021

1. Уяснить назначение создаваемой программной системы и определить потенциальных пользователей.
2. Сформировать первичные и детальные требования к ПС.
3. Перечислить задачи архитектуры ПО, важные для заказчика.
4. Рассмотреть ключевые вопросы проектирования, оказывающие существенное влияние на архитектуру ПС (параллелизм вычислений, асинхронность коротких и длинных процессов, обеспечение отказоустойчивости и др.).
5. Выбрать методологию разработки ПО (структурный или объектно-ориентированный подход).
6. Определить цели архитектуры.
7. Выявит основные сценарии через варианты использования.
8. Определить тип приложения.
9. Определить ограничения развёртывания.
10. Используя архитектурные системные паттерны, проанализировать и выбрать архитектурный стиль проектирования.
11. Изобразить виды архитектуры ПС с помощью UML-диаграмм.
12. Рассмотреть возможность паттерн-ориентированного проектирования ПС.
13. Спроектировать простой пользовательский интерфейс.
14. Выбрать необходимые инструментальные средства разработки ПС (языки и среды программирования).
15. Реализовать несколько базовых модулей ПС.
16. Проанализировать подходы к оцениванию архитектуры.

График выполнения курсового проекта:

|  |  |
| --- | --- |
| Неделя | Работа |
| 1 | Выдача задания и изучение задания |
| 2-3 | Анализ требований к системе и способов их реализации |
| 4-5 | Проектирование архитектуры системы |
| 6 | Оценка спроектированной системы |
| 7-8 | Описание и проектирование паттернов |
| 9-13 | Программная реализация модуля |
| 14 | Оформление пояснительной записки |
| 15-17 | Защита курсового проекта |

Дата выдачи задания \_\_.09.2021

Руководители:

Ассистент кафедры ПИ им.Л.П. Фельдмана\_\_\_\_\_\_\_\_\_Серёженко О.А.

должность, звание подпись расшифровка подписи

Ассистент кафедры ПИ им.Л.П. Фельдмана\_\_\_\_\_\_\_\_\_Ищенко А.П.

должность, звание подпись расшифровка подписи

Студент ПИ-19а \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Саевский О.В.

номер группы подпись расшифровка подписи

# ПРИЛОЖЕНИЕ Б

Функциональная модель

Функциональная модель системы представлена на рисунке Б.1.

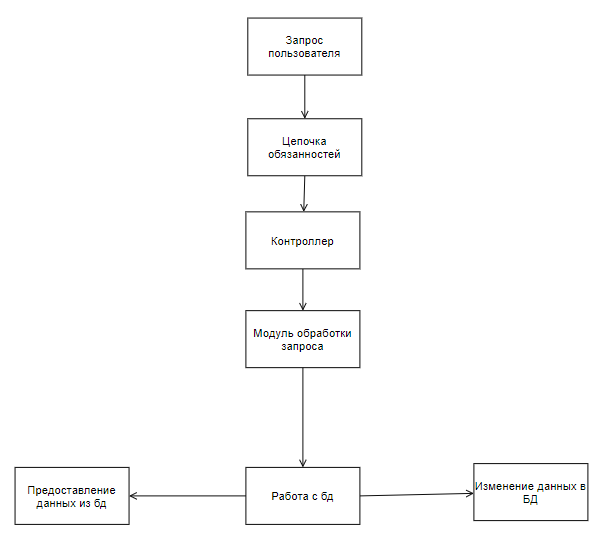


Рисунок Б.1 – Функциональная модель системы

# ПРИЛОЖЕНИЕ В

Листинг программы

namespace Common.Models

{

public class City

{

public int Id { get; set; }

public string Name { get; set; }

}

}

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllersWithViews();

builder.Services.AddTransient<PostreSQL>();

builder.Services.AddTransient<ServiceRepository>();

builder.Services.AddTransient<CheckServiceInDbMiddleware>();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (!app.Environment.IsDevelopment())

{

app.UseExceptionHandler("/Home/Error");

// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();

namespace Common.Models

{

public class Order

{

public int Id { get; set; }

public int UserId { get; set; }

public int TaskId { get; set; }

public ServiceTask Task { get; set; }

}

}

namespace Common.Models

{

public class Service

{

public int Id { get; set; }

public string Name { get; set; }

public virtual ICollection<ServiceTask> Tasks { get; set; }

}

}

namespace Common.Models

{

public class ServiceTask

{

public int Id { get; set; }

public string Name { get; set; }

public double Price { get; set; }

public virtual ICollection<Service> Services { get; set; }

}

}

using Common.Models;

using DatabaseServer.Repositories;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

namespace DatabaseServer.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class OrderController : ControllerBase

{

private readonly OrderRepository \_orderRepository;

public OrderController(OrderRepository orderRepository)

{

\_orderRepository = orderRepository;

}

[HttpGet]

public IActionResult Get(int user\_id)

{

var orders = Enumerable.Empty<Order>();

if (user\_id == 0)

orders = \_orderRepository.GetAll();

else orders = \_orderRepository.GetAll(user\_id);

return Ok(orders);

}

[Route ("add")]

[HttpPost]

public async Task<IActionResult> Add([FromBody] Order order)

{

try

{

await \_orderRepository.Add(order);

await \_orderRepository.SaveAsync();

return Ok();

}

catch (Exception)

{

return BadRequest();

}

}

}

}

using Common.Models;

using DatabaseServer.Repositories;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

namespace DatabaseServer.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class ServiceController : ControllerBase

{

private readonly ServiceRepository \_serviceRepository;

public ServiceController(ServiceRepository serviceRepository)

{

\_serviceRepository = serviceRepository;

}

[HttpGet]

public IActionResult Get()

{

var services = \_serviceRepository.GetAll();

return Ok(services);

}

[Route("add")]

[HttpPost]

public async Task<IActionResult> Post([FromBody]Service service)

{

try

{

await \_serviceRepository.Add(service);

await \_serviceRepository.SaveAsync();

return Ok();

}

catch (Exception)

{

return BadRequest();

}

}

}

}

using Common.Models;

using DatabaseServer.Repositories;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

namespace DatabaseServer.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class TaskController : ControllerBase

{

private readonly ServiceTaskRepository \_serviceTaskRepository;

public TaskController(ServiceTaskRepository serviceTaskRepository)

{

\_serviceTaskRepository = serviceTaskRepository;

}

[Route("/api/task/")]

[HttpGet]

public IActionResult Get()

{

var tasks = \_serviceTaskRepository.GetAll().ToList();

return Ok(tasks);

}

[Route("/api/task/id={id}")]

[HttpGet]

public IActionResult GetById(int id)

{

var task = \_serviceTaskRepository.Get(id);

return Ok(task);

}

[Route("/api/task/name={name}")]

[HttpGet]

public IActionResult GetByName(string name)

{

var task = \_serviceTaskRepository.Get(name);

return Ok(task);

}

[Route ("add")]

[HttpPost]

public async Task<IActionResult> Add([FromBody] ServiceTask task)

{

try

{

await \_serviceTaskRepository.Add(task);

await \_serviceTaskRepository.SaveAsync();

return Ok();

}

catch (Exception)

{

return BadRequest();

}

}

}

}

using Common.Models;

using Microsoft.EntityFrameworkCore;

#pragma warning disable CS8618

namespace DatabaseServer.Database

{

public class PostgresConnection : DbContext

{

public PostgresConnection(DbContextOptions<PostgresConnection> options) : base(options)

{

//Database.EnsureDeleted();

Database.EnsureCreated();

}

public DbSet<Order> Orders { get; set; }

public DbSet<Service> Services { get; set; }

public DbSet<ServiceTask> Tasks { get; set; }

public DbSet<City> Cities { get; set; }

}

}

using Common.Models;

using DatabaseServer.Database;

namespace DatabaseServer.Repositories

{

public class CityRepository : IRepository<City>

{

private readonly PostgresConnection \_db;

public CityRepository(PostgresConnection db)

{

\_db = db;

}

public async Task Add(City obj)

{

await \_db.Cities.AddAsync(obj);

}

public Task Delete(City obj)

{

\_db.Cities.Remove(obj);

return Task.CompletedTask;

}

public async Task Dispose()

{

await \_db.DisposeAsync();

}

public City? Get(int id)

{

return \_db.Cities.SingleOrDefault(x=>x.Id == id);

}

public IEnumerable<City> GetAll()

{

return \_db.Cities.AsEnumerable();

}

public async Task SaveAsync()

{

await \_db.SaveChangesAsync();

}

}

}

namespace DatabaseServer.Repositories

{

public interface IRepository <T> where T : class

{

public IEnumerable<T> GetAll();

public T? Get(int id);

public Task Add(T obj);

public Task Delete(T obj);

public Task Dispose();

public Task SaveAsync();

}

}

using Common.Models;

using DatabaseServer.Database;

using Microsoft.EntityFrameworkCore;

namespace DatabaseServer.Repositories

{

public class OrderRepository : IRepository<Order>

{

private readonly PostgresConnection \_db;

public OrderRepository(PostgresConnection db)

{

\_db = db;

}

public async Task Add(Order obj)

{

await \_db.Orders.AddAsync(obj);

}

public Task Delete(Order obj)

{

throw new NotImplementedException();

}

public async Task Dispose()

{

await \_db.DisposeAsync();

}

public Order? Get(int id)

{

throw new NotImplementedException();

}

public IEnumerable<Order> GetAll()

{

return \_db.Orders.Include(o=>o.Task).AsEnumerable();

}

public IEnumerable<Order> GetAll(int userId)

{

return \_db.Orders.Include(o => o.Task).Where(x=>x.UserId == userId).AsEnumerable();

}

public async Task SaveAsync()

{

await \_db.SaveChangesAsync();

}

}

}

using Common.Models;

using DatabaseServer.Database;

namespace DatabaseServer.Repositories

{

public class ServiceRepository : IRepository<Service>

{

private readonly PostgresConnection \_db;

public ServiceRepository(PostgresConnection db)

{

\_db = db;

}

public async Task Add(Service obj)

{

await \_db.Services.AddAsync(obj);

}

public Task Delete(Service obj)

{

throw new NotImplementedException();

}

public Task Dispose()

{

throw new NotImplementedException();

}

public Service? Get(int id)

{

throw new NotImplementedException();

}

public IEnumerable<Service> GetAll()

{

return \_db.Services.AsEnumerable();

}

public async Task SaveAsync()

{

await \_db.SaveChangesAsync();

}

}

}

using Common.Models;

using DatabaseServer.Database;

namespace DatabaseServer.Repositories

{

public class ServiceTaskRepository : IRepository<ServiceTask>

{

private readonly PostgresConnection \_db;

public ServiceTaskRepository(PostgresConnection db)

{

\_db = db;

}

public async Task Add(ServiceTask obj)

{

await \_db.Tasks.AddAsync(obj);

}

public Task Delete(ServiceTask obj)

{

throw new NotImplementedException();

}

public async Task Dispose()

{

await \_db.DisposeAsync();

}

public ServiceTask? Get(int id)

{

return \_db.Tasks.FirstOrDefault(x => x.Id == id);

}

public ServiceTask? Get(string name)

{

return \_db.Tasks.FirstOrDefault(x => x.Name == name);

}

public IEnumerable<ServiceTask> GetAll()

{

return \_db.Tasks.AsEnumerable();

}

public async Task SaveAsync()

{

await \_db.SaveChangesAsync();

}

}

}