РОСЖЕЛДОР

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Ростовский государственный университет путей сообщения»

(ФГБОУ ВО РГУПС)

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

Кафедра «ВТ и АСУ»

«Объектно-ориентированная разработка программ»

Пояснительная записка

к курсовой работе по дисциплине

«Технологии программирования»

ТП 01.13 ПЗ

Учебная группа АВБ-4-032

Выполнил студент Мазуров И.А.

(подпись студента)

Руководитель курсовой работы \_\_\_\_\_\_\_\_\_\_\_ Ведерникова О.Г.

(подпись)

Работа допущена к защите\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

Работа защищена \_\_\_\_\_\_\_\_\_с оценкой \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата) (подпись руководителя)

г. Ростов – на – Дону

2022 г.

**Оглавление**

[Введение 3](#_Toc98832338)

[1 Словесное описание 4](#_Toc98832339)

[2 Диаграмма вариантов использования 5](#_Toc98832340)

[3 Диаграмма классов 6](#_Toc98832341)

[4 Сценарий вариантов использования 7](#_Toc98832342)

[5 Диаграмма последовательностей 9](#_Toc98832343)

[6 Оценка проекта на основе LOC- и FP-метрик 11](#_Toc98832344)

[6.1 Введение 11](#_Toc98832345)

[6.2 Расчет FP-оценки 14](#_Toc98832346)

[6.3 Расчет LOC-оценки 18](#_Toc98832347)

[6.4 Расчет COCOMO-оценки 20](#_Toc98832348)

[7 Реализация некоторого модуля 23](#_Toc98832349)

[Заключение 26](#_Toc98832350)

[Список использованных источников 27](#_Toc98832351)

# **Введение**

Сервис-ориентированная архитектура – модульный подход к разработке программного обеспечения, базирующийся на обеспечении удаленного по стандартизированным протоколам использования распределённых, слабо связанных легко заменяемых компонентов (сервисов) со стандартизированными интерфейсами.

Интерфейсы компонентов в сервис-ориентированной архитектуре инкапсулируют детали реализации (операционную систему, платформу, язык программирования) от остальных компонентов, таким образом обеспечивая комбинирование и многократное использование компонентов для построения сложных распределённых программных комплексов, обеспечивая независимость от используемых платформ и инструментов разработки, способствуя масштабируемости и управляемости создаваемых систем.

Для проектирования такого сложного ПО удобно использовать унифицированный язык проектирования UML. Он позволяет описать систему практически со всех возможных сторон, определив различные аспекты поведения системы. Важным является то, что проектирование на языке UML в различных прикладных средах идентичны, что указывает на универсальность проектируемых средств.

UML объектно-ориентирован, в следствие чего методы проектирования семантически близки к методам программирования на современных языках поддерживающих ООП.

Каждая диаграмма UML конкретизирует различные представления о модели системы. При этом, диаграмма вариантов использования представляет концептуальную модель системы, которая является исходной для построения всех остальных диаграмм. Диаграмма классов является логической моделью, отражающей статические аспекты структурного построения системы, а диаграмма последовательностей, также являющаяся разновидностью логической модели, отражает динамические аспекты её функционирования.

# **Словесное описание**

Проектирование системы приложения для организации работы контакт-центра.

Приложение является основной платформой для работы операциониста любого контакт-центра и предоставляет различные сервисы, такие как:

1. Рассылка сообщений выбранным номерам
2. Исходящий звонок по конкретному номеру
3. Ответ на входящий звонок

Для этого приложения предусмотрен один актер – авторизованный пользователь.

Для авторизированного пользователя доступен следующий функционал:

1. Окно авторизации
   1. Авторизация по логину и паролю
   2. Выход из приложения
2. Главный экран
   1. Отправка сообщений
   2. Совершение звонка
   3. Ответ на входящий звонок
   4. Выход из приложения
3. Просмотр статистики
4. Создание отчета за указанный период

# **Диаграмма вариантов использования**

Моделирование в UML можно представить как  
некоторый процесс поуровневого спуска от наиболее  
обшей и абстрактной концептуальной модели  
исходной системы к логической, а затем и к  
физической модели, соответствующей программной  
системы. Для достижения этих целей вначале строится  
модель в форме так называемой диаграммы вариантов  
использования (use case diagram), которая описывает  
функциональное назначение системы или, другими  
словами, то, что система будет делать в процессе  
своего функционирования. Диаграмма вариантов  
использования является исходным концептуальным  
представлением или концептуальной моделью  
системы в процессе ее проектирования и разработки.

Описание функционала диаграммы вариантов использования для проектируемого проекта представлена в пункте 1.

Выполненная диаграмма вариантов использования представлена на рис. 2.1:

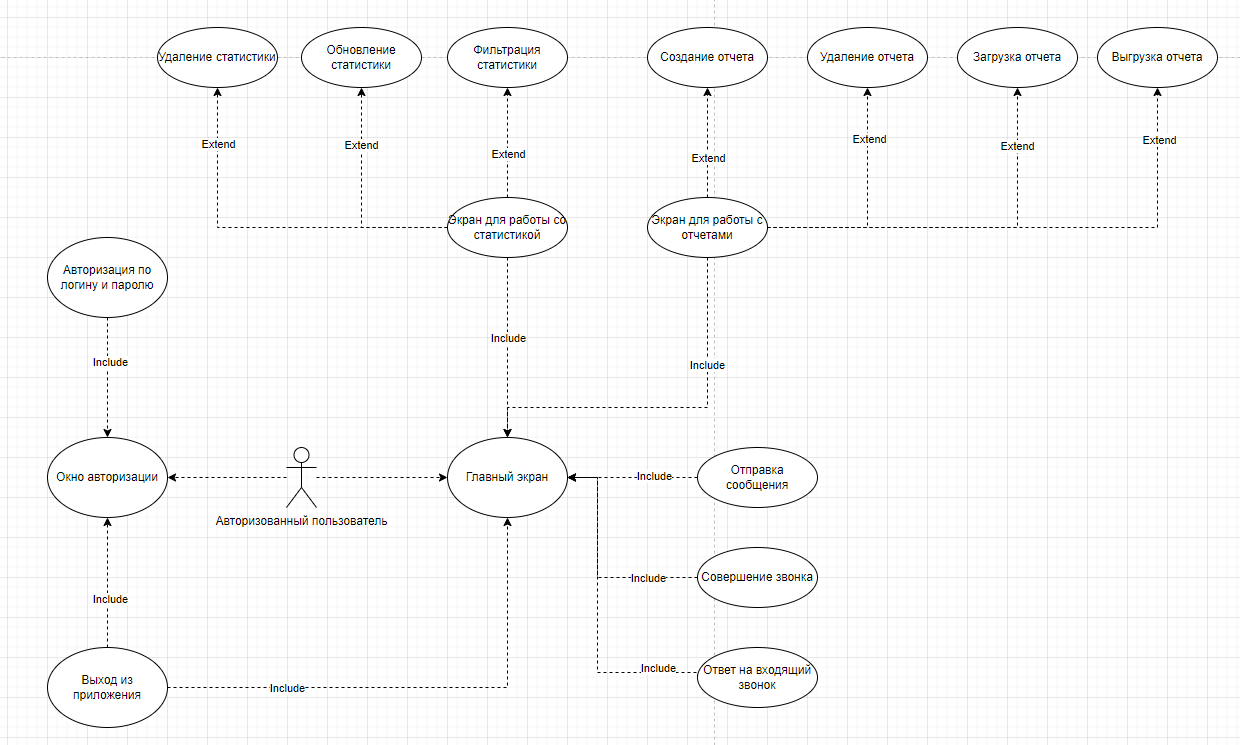


Рисунок 2.1 – Диаграмма вариантов использования

# **Диаграмма классов**

Диаграмма классов (англ. class diagram) — структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

При представлении сущностей реального мира разработчику требуется отразить их текущее состояние, их поведение и их взаимные отношения. На каждом этапе осуществляется абстрагирование от концепций, которые не относятся к реальности (производительность, инкапсуляция, видимость и т. п.). Классы можно рассматривать с позиции различных уровней, выделяют три основных: аналитический уровень, уровень проектирования и уровень реализации.

Описание классов для проектируемого проекта представлена в пункте 1.

Выполненная диаграмма классов представлена на рис. 3.1:

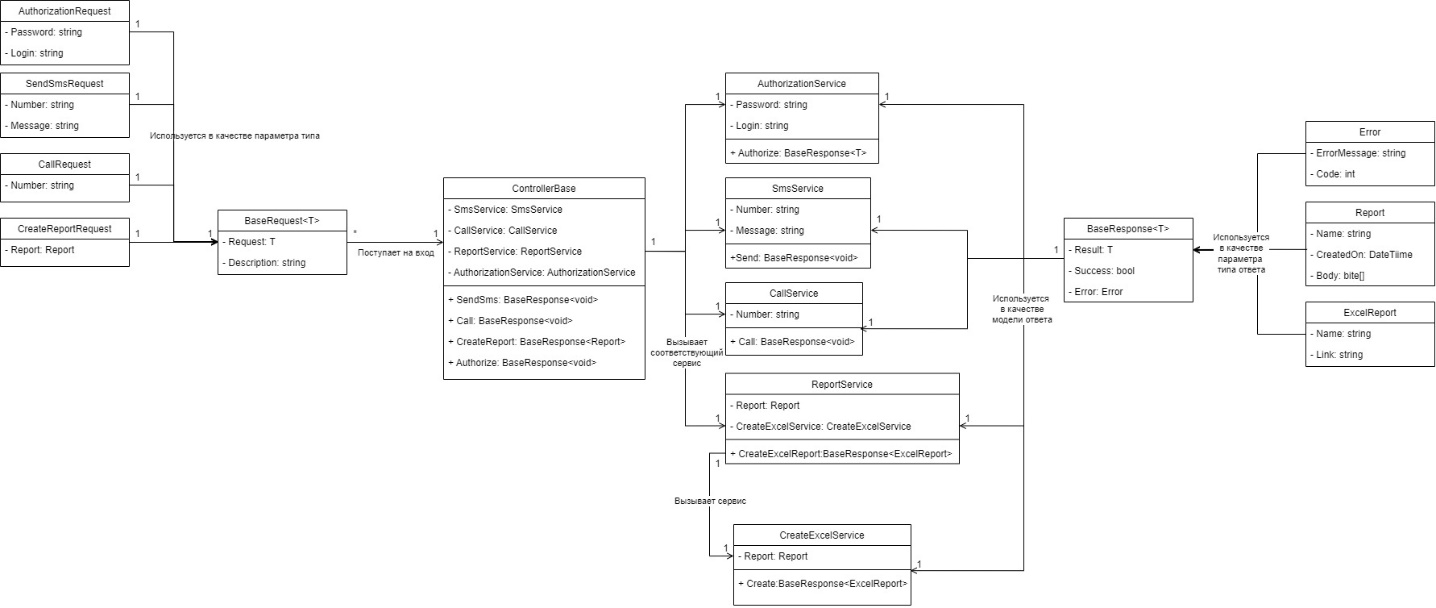


Рисунок 3.1 – Диаграмма Классов

# **Сценарий вариантов использования**

Сценарий №1выполнения ВИ «Отправка сообщения»

Главный раздел

Вариант использования: Отправка сообщения.

Актеры: Пользователь, Система.

Цель: отправить сообщение.

Предусловие: Профиль пользователя не авторизован.

Краткое описание: Пользователь пользуется своим профилем для отправки сообщения. Пользователь проходит авторизацию в системе, далее запускает необходимую процедуру после чего система совершает отправку письма по указанному номеру.

Тип: Базовый

Ссылки на другие варианты использования:

Включает в себя ВИ авторизации профиля пользователя.

Раздел – «Типичный ход событий»:

Действия актеров:

1. Пользователь попадает в Окно Авторизации.
2. Пользователь вводит свои данные для авторизации (логирования) в соответствующие поля приложения.
3. Система создает подключение к БД для того, чтобы сверить введённые данные с имеющимися в БД.

Исключение №1: Ошибка авторизации, не удалось создать подключение к БД.

1. Система сверяет введенные данные с имеющимися в БД.

Исключение №2: Ошибка авторизации, введенные данные не найдены в БД.

1. Система выдает сообщение пользователю о успешной авторизации.
2. Пользователь вводит номер телефона и текст сообщения и кликает на соответствующую кнопку для его отправки.
3. Система отправляет сообщение.

Исключение №3: При отправке сообщения произошла ошибка, которая не дает возможности совершить отправку.

1. Система выводит на экран пользователя сообщение об успешной отправке.
2. Система готова к дальнейшему использованию.

Раздел – «Исключения»:

Действия актеров:

Исключение №1: Ошибка авторизации, не удалось создать подключение к БД.

1. Система выдаёт сообщение о соответствующей ошибке и возвращает пользователя на шаг 2 (с сохранением введенных данных).
2. Пользователь изменяет свои данные для авторизации (логирования) в соответствующих полях приложения.

Исключение №2: Ошибка авторизации, введенные данные не найдены в БД.

1. Система выдаёт сообщение о соответствующей ошибке и возвращает пользователя на шаг 2 (с сохранением введенных данных).
2. Пользователь изменяет свои данные для авторизации (логирования) в соответствующих полях приложения.

Исключение №3: При отправке сообщения произошла ошибка, которая не дает возможности совершить отправку.

1. Система выдаёт сообщение о соответствующей ошибке и возвращает пользователя на шаг 6.
2. Пользователь исправляет ошибку и снова отправляет сообщение.

# **Диаграмма последовательностей**

Диаграммы последовательности (sequence diagram) являются видом диаграмм взаимодействия языка UML, которые описывают отношения объектов в различных условиях. Условия взаимодействия задаются сценарием, полученным на этапе разработки диаграмм вариантов использования.

Описание классов для проектируемого проекта представлена в пункте 1.

Выполненная диаграмма последовательностей представлена на рис. 5.1:

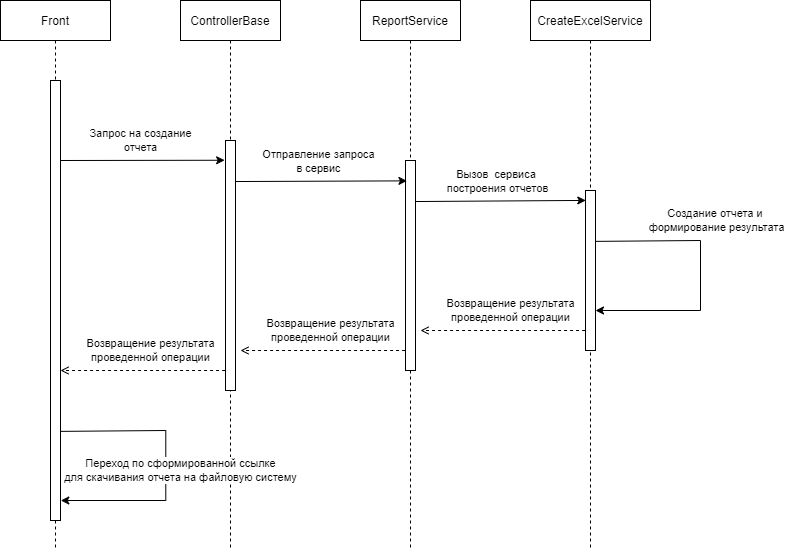


Рисунок 5.1 – Диаграмма последовательностей

На рисунке 5.1 изображена диаграмма последовательности, на которой продемонстрирован процесс взаимодействия класса-контроллера с классами сервисами при осуществлении запроса на создание отчета.

Из диаграммы видно, что инициатором осуществления процессов является пользователь, использующий веб-интерфейс. Кроме того, видно, что запрос обрабатывается три разными классами. Это сделано для того, чтобы бизнес-логику от технической реализации создания отчета. После получения результата работы программы происходит автоматическое скачивание файла.

Таким образом, в результате построения диаграммы последовательности разработаны подробные описания процессов внутри приложения, которые будут отражать функциональную ценность информационной системы.

# **Оценка проекта на основе LOC- и FP-метрик**

## Введение

Цель этой деятельности - сформировать предварительные оценки, которые позволят:

* предъявить заказчику корректные требования по стоимости и затратам на разработку программного продукта;
* составить план программного проекта.

В оценке стоимости ПО используются 2-е единицы оценки: функциональная точка Function Point (FP) и строка кода Line of Code (LOC).

При расчете FP и LOC оценок необходимо выполнить анализ предварительной трудоемкости и стоимости разработки программы.

К достоинствам функционально-ориентированных метрик относят:

1. Не зависят от языка программирования.

2. Легко вычисляются на любой стадии проекта.

К недостаткам функционально-ориентированных метрик относят: 1.Результаты, основанные на субъективных данных, используются не прямые, а косвенные измерения.

2. FP-оценки легко пересчитать в LOC-оценки. Результаты пересчета зависят от языка программирования, используемого для реализации ПО.

Используется 5 информационных характеристик описание которых представлено ниже, а их оценка в таблице 1.

1. Количество внешних вводов. Подсчитываются все вводы пользователя, по которым поступают разные прикладные данные. Вводы должны быть отделены от запросов, которые подсчитываются отдельно.

2. Количество внешних выводов. Подсчитываются все выводы, по которым к пользователю поступают результаты, вычисленные программным приложением. В этом контексте выводы означают отчеты, экраны, распечатки, сообщения об ошибках. Индивидуальные единицы данных внутри отчета отдельно не подсчитываются.

3. Количество внешних запросов. Под запросом понимается диалоговый ввод, который приводит к немедленному программному ответу в форме диалогового вывода. При этом диалоговый ввод в приложении не сохраняется, а диалоговый вывод не требует выполнения вычислений. Подсчитываются все запросы - каждый учитывается отдельно.

4. Количество внутренних логических файлов. Подсчитываются все логические файлы (то есть логические группы данных, которые могут быть частью базы данных или отдельным файлом).

5. Количество внешних интерфейсных файлов. Подсчитываются все логические файлы из других приложений, на которые ссылается данное приложение.

Каждой из выявленных характеристик ставится в соответствие сложность. Для этого характеристике назначается низкий, средний или высокий ранг, а затем формируется числовая оценка ранга.

Данные для определения ранга и оценки сложности транзакций и файлов приведены в таблицах:

Таблица 1.1 – Ранг и оценка сложности внешних вводов

|  |  |  |  |
| --- | --- | --- | --- |
| Ссылки на файлы | Элементы данных | | |
|  | 1-4 | 5-15 | >15 |
| 0-1 | Низкий(3) | Низкий(3) | Средний(4) |
| 2 | Низкий(3) | Средний(4) | Высокий(6) |
| >2 | Средний(4) | Высокий(6) | Высокий(6) |

Таблица 1.2 – Ранг и оценка сложности внешних выводов

|  |  |  |  |
| --- | --- | --- | --- |
| Ссылки на файлы | Элементы данных | | |
|  | 1-4 | 5-19 | >19 |
| 0-1 | Низкий(4) | Низкий(4) | Средний(5) |
| 2-3 | Низкий(4) | Средний(5) | Высокий(7) |
| >3 | Средний(5) | Высокий(7) | Высокий(7) |

Таблица 1.3 – Ранг и оценка сложности внешних запросов

|  |  |  |  |
| --- | --- | --- | --- |
| Ссылки на файлы | Элементы данных | | |
|  | 1-4 | 5-19 | >19 |
| 0-1 | Низкий(3) | Низкий(3) | Средний(4) |
| 2-3 | Низкий(3) | Средний(4) | Высокий(6) |
| >3 | Средний(4) | Высокий(6) | Высокий(6) |

Таблица 1.4 – Ранг и оценка сложности внутренних логических файлов

|  |  |  |  |
| --- | --- | --- | --- |
| Типы элементов записей | Элементы данных | | |
|  | 1-19 | 20-50 | >50 |
| 1 | Низкий(7) | Низкий(7) | Средний(10) |
| 2-5 | Низкий(7) | Средний(10) | Высокий(15) |
| >5 | Средний(10) | Высокий(15) | Высокий(15) |

Таблица 1.5 – Ранг и оценка сложности внешних интерфейсных файлов

|  |  |  |  |
| --- | --- | --- | --- |
| Ссылки на файлы | Элементы данных | | |
|  | 1-19 | 20-50 | >50 |
| 1 | Низкий(5) | Низкий(5) | Средний(7) |
| 2-5 | Низкий(5) | Средний(7) | Высокий(10) |
| >5 | Средний(7) | Высокий(10) | Высокий(10) |

## Расчет FP-оценки

Таблица 2.1 – Исходные данные для расчета FP-оценки

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя характеристики | Ранг, количество, сложность | | | |
| Низкий | Средний | Высокий | Итого |
| Внешние вводы | 0\*3=0 | 0\*4=20 | 10\*6=0 | 60 |
| Внешние выводы | 0\*4=0 | 0\*5=0 | 10\*7=70 | 70 |
| Внешние запросы | 1\*3=3 | 0\*4=0 | 0\*6=0 | 3 |
| Внутренние логические файлы | 0\*7=0 | 35\*10=350 | 0\*15=0 | 350 |
| Внешние интерфейсные файлы | 0\*5=0 | 0\*7=0 | 0\*10=0 | 0 |
| Всего | | | | 483 |

Где:

ранг показан в 2-м столбце 2-й строке (низкий, средний и т.д.);

количество отображено в 1-м параметре в столбцах ранга;

сложность отображена во 2-м параметре в столбцах ранга.

Ранг определяется, по количественной оценке, характеристики (кол-во внешних вводов) и по кол-ву ссылок на внутренние логические или внешние интерфейсные файлы, а также по кол-ву типов элементов-записей в файлах и их типах. Кол-во по количеству. Сложность – константное значение.

Кол-во функциональных указателей вычисляется по формуле:

где Fi – коэффициенты регулировки сложности в диапазоне:

* 0 – не влияет;
* 1 – случайное влияние;
* 2 – небольшое влияние;
* 3 – среднее влияние;
* 4 – важное влияние;
* 5 – основное влияние;

Значение коэффициентов регулировки сложности представлены в таблице 2.2.

Таблица 2.2 – Значение коэффициентов регулировки сложности

|  |  |  |  |
| --- | --- | --- | --- |
| № | Системный параметр | Описание | Коэффициент |
| 1 | Передача данных | Сколько средств связи требуется для передачи или обмена информацией, или с приложением или системой? | 4 |
| 2 | Распределенная обработка данных | Как обрабатываются распределенные данные и функции обработки? | 3 |
| 3 | Производительность | Нуждается пользователь в фиксации времени ответа или производительности? | 4 |
| 4 | Распространённость используемой конфигурации | Насколько распространена текущая аппаратная платформа, на которой будет выполняться приложение? | 4 |
| 5 | Скорость транзакций | Как часто выполняются транзакции? (каждый день, каждую неделю, каждый месяц) | 4 |
| 6 | Оперативный ввода данных | Какой процент информации надо вводить в режиме онлайн? | 3 |
| 7 | Эффективность работы конечного пользователя | Приложение проектировалось для обеспечения эффективной работы конечного пользователя? | 5 |
| 8 | Оперативное обновление | Как много внутренних файлов обновляется в онлайновой транзакции? | 4 |
| 9 | Сложность обработки | Выполняет ли приложение интенсивную логическую или математическую обработку? | 1 |
| 10 | Повторная используемость | Приложение разрабатывалось для удовлетворения требований одного или многих пользователей? | 1 |
| 11 | Легкость инсталляции | Насколько трудны преобразование и инсталляция приложения? | 1 |
| 12 | Легкость эксплуатации | Насколько эффективны и/или автоматизированы процедуры запуска, резервирования и восстановления? | 3 |
| 13 | Разнообразные условия размещения | Была ли спроектирована, разработана и поддержана возможность инсталляции приложения в разных местах для различных организаций? | 4 |
| 14 | Простота изменений | Была ли спроектирована, разработана и поддержана в приложении простота изменений? | 3 |
| Всего | | | 44 |

Теперь можно рассчитать количество функциональных указателей:

Так как это количество функциональных указателей, округляем его до ближайшего целого и получаем 527 функциональных точек.

## Расчет LOC-оценки

Line of Code (LOC) – это оценка ПО, используемая для измерения его объёма с помощью подсчёта количества строк в тексте исходного кода. К преимуществам использования LOC, как единицы размера ПО, относят простоту, а недостатками является следующее:

* размер проекта в LOC может быть определен только после его завершения;
* LOC зависит от языка программирования;
* LOC не учитывает качества кода.

Зная кол-во функциональных указателей, можем получить число строк кода. В языке C# одна функциональная точка примерно равна 53 строкам кода (Таблица 3.1).

Таблица 3.1 – Кол-во операторов на один FP

|  |  |
| --- | --- |
| Язык программирования | Кол-во операторов на один FP |
| Assembler | 320 |
| C | 128 |
| Cobol | 106 |
| Fortran | 106 |
| Pascal | 90 |
| C++ | 64 |
| LISP | 64 |
| Prolog | 64 |
| C# | 53 |
| Java | 53 |
| Ada 95 | 49 |
| Visual Basic | 32 |
| Visual C++ | 34 |
| Smalltalk | 22 |
| Perl | 21 |

Формула для пересчета из FP в LOC:

Получим, что число строк кода равно:

строк кода.

## Расчет COCOMO-оценки

COnstructive COst MOdel (COCOMO – модель издержек разработки) – это алгоритмическая модель оценки стоимости разработки программного обеспечения.

Основное уравнение этой модели имеет вид:

где:

* А – масштабный коэффициент. А = 2.5;
* Размер – размер ПО выраженный в тыс. LOC;
* Me – множитель поправки зависит от 7-и формирователей затрат, характеризирующих продукт, процесс и персонал (Таблица 4.1);
* Показатель степени B отражает нелинейную зависимость затрат от размера проекта (от длины кода LOC).

Значение показателя степени B измеряется в диапазоне [1.01,…,1.26], зависит от 5-и масштабных факторов Wi и вычисляется по формуле:

.

Общая характеристика масштабных факторов Wi приведена в таблице 4.1.

Таблица 4.1 – Характеристика масштабных факторов Wi

|  |  |  |
| --- | --- | --- |
| Масштабный фактор Wi | Пояснение | Wi |
| 1 Предсказуемость, наличие прецедентов | Отражает предыдущий опыт организации в реализации проектов этого типа. Очень низкий (=5) означает отсутствие опыта. Сверхвысокий (=0) означает, что организация полностью знакома с этой прикладной областью | 3-среднее |
| 2 Гибкость разработки | Отражает степень гибкости процесса разработки. Очень низкий (=5) означает, что используется заданный процесс. Сверхвысокий (=0) означает, что клиент установил только общие цели | 3-среднее |
| 3 Разрешение рисков в архитектуре | Отражает степень выполняемого анализа риска. Очень низкий (=5) означает малый анализ. Сверхвысокий (=0) означает полный и сквозной анализ риска | 4-среднее |
| 4 Связность группы | Отражает, насколько хорошо разработчики группы знают друг друга, и насколько удачно они совместно работают. Очень низкий (=5) означает очень трудные взаимодействия. Сверхвысокий (=0) означает интегрированную группу без проблем взаимодействия | 1-очень высокий |
| 5 Зрелость процесса | Означает зрелость процесса в организации. Вычисление этого фактора может выполняться по вопроснику CMM | 3-среднее |
| Всего | | 14 |

Теперь вычислим показатель B:

Множитель поправки Ме зависит от набора формирователей затрат *ЕМi* перечисленных в таблице 4.2.

Таблица 4.2 – Формирователи затрат *EMi* для раннего этапа проектирования

|  |  |  |
| --- | --- | --- |
| Обозначение | Название | *EMi* |
| PERS | Возможности (способности) персонала | Среднее=1 |
| RCPX | Надежность и сложность продукта | Среднее=1 |
| RUSE | Требуемое повторное использование | Среднее=1 |
| PDIF | Трудность (сложность) платформы | Среднее=1 |
| PREX | Опытность персонала | Среднее=1 |
| FCIL | Средства поддержки | Среднее=1 |
| SCED | Сроки | Среднее=1 |

Для расчета множителя поправки, зависящего от 7-и формирователей затрат, характеризующих продукт, процесс и персонал, воспользуемся формулой:

.

.

Для расчета затрат на разрабатываемый продукт, воспользуемся формулой, составные части которой мы уже рассчитали:

Исходя из расчетов, один человек может выполнить поставленную задачу за 7 с половиной месяцев.

# **Реализация модуля получения http-запросов ControllerBase**

В ходе выполнения курсовой работы был разработан модуль получения http-запросов ControllerBase:

using CallCenter.Models;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace CallCenter.Controller

{

public class ControllerBase

{

private readonly ILogger<ControllerBase> logger;

private readonly IAuthorizationService authorizationService;

private readonly ISmsService smsService;

private readonly ICallService callService;

private readonly IReportService reportService;

public ControllerBase(IAuthorizationService authorizationService,

ISmsService smsService,

ICallService callService,

IReportService reportService,

ILogger<ControllerBase> logger)

{

this.authorizationService = authorizationService;

this.smsService = smsService;

this.callService = callService;

this.reportService = reportService;

this.logger = logger;

}

[HttpGet]

public BaseResponse Authorization(BaseRequest<AuthorizationRequest> request)

{

return authorizationService.Authorize(request);

}

[HttpGet]

public BaseResponse SendSms(BaseRequest<SendSmsRequest> request)

{

return smsService.Send(request);

}

[HttpGet]

public BaseResponse CreateReport(BaseRequest<CreateReportRequest> request)

{

return smsService.Create(request);

}

}

}

namespace CallCenter.Models

{

public class BaseRequest<T>

{

public T Request { get; set; }

public string Description { get; set; }

}

public class BaseResponse<T>

{

public T Result { get; set; }

public bool Success { get; set; }

public Error Error { get; set; }

}

public class BaseResponse

{

public bool Success { get; set; }

public Error Error { get; set; }

}

public class Error

{

public string Message { get; set; }

public int Code { get; set; }

}

public class AuthorizationRequest

{

public string Password { get; set; }

public string Login { get; set; }

}

public class SendSmsRequest

{

public string Number { get; set; }

public string Message { get; set; }

}

public class CreateReportRequest

{

public Report Report { get; set; }

}

public class Report

{

public string Name { get; set; }

public DateTime CreatedOn { get; set; }

}

}

# **Заключение**

Целью данной курсовой работы является «Проектирование приложения для организации работы контакт-центра».

Сформировано словесное описание предметной области и функционала приложения. Разработаны основные диаграммы на языке UML, такие как диаграмма вариантов использования, диаграмма классов и диаграмма последовательностей. Создан сценарий ВИ. Проведены предварительные расчеты по трудоемкости разработки приложения, которые составили 7,5 месяцев на одного человека. Разработан один из модулей приложения, а именно модуль получения запросов.

На основе полученных расчетов и спроектированных диаграмм можно разработать функционирующее приложение для организации работы контакт-центра.

# **Список использованных источников**

1. **Ларман К.** Применение UML 2.0 и шаблонов проектирования / К. Ларман. – Вильямс, 2019. – 736 с.
2. **Галиаскаров Э.Г.** Анализ и проектирование систем с использованием UML / Э.Г. Галиаскаров, А.С. Воробьёв – СПб.: Питер, 2022. – 126.
3. **Таненбаум Д.** Архитектура компьютера. 6–е изд. – СПб.: Питер, 2019. – 816с.
4. **Коваленко В.В.** Проектирование информационных систем. Учебное пособие. Студентам ВУЗов / В.В. Коваленко – Форум, 2021 г. – 357 ст.
5. **Вейцман В.М.** Проектирование информационных систем. Учебное пособие. СПО / В.М. Вейцман – Лань, 2020 г. – 316 ст.
6. **Гома Х.** UML. Проектирование систем реального времени, распределенных и параллельных приложений / Х. Гома – Litres, 2022г. – 316 ст.