

**ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ И НАУКИ ГОРОДА МОСКВЫ
ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ГОРОДА МОСКВЫ
«ПЕРВЫЙ МОСКОВСКИЙ ОБРАЗОВАТЕЛЬНЫЙ КОМПЛЕКС»
(ГБПОУ «1-й МОК»)**

Допустить к защите:

Заведующая отделением среднего
профессионального образования
факультета «Информационные
технологии и управление»

_____/Пашохонова Е.Е.

«___» _____ 2022 г

ДИПЛОМНЫЙ ПРОЕКТ

по специальности

09.02.07 «Информационные системы и программирование»

(наименование специальности)

Тема: «Разработка программы планирования распорядка дня, для
руководителя ресторана»

Обучающийся Кулаков Тимур Алексеевич группы 41 ИС
(фамилия, имя, отчество)

Руководитель Зуев Денис Иванович
(фамилия, имя, отчество)

Работа защищена «___» июня 2022 года с оценкой _____

Секретарь Государственной
экзаменационной комиссии

(подпись)

Москва, 2022

Содержание

Введение	4
1 Анализ предметной области	6
1.1 Описание предметной области	6
1.2 Организационная структура организации	9
1.3 Контекстная диаграмма	9
1.4 Диаграмма декомпозиции	11
2 Проектирование информационной подсистемы	13
2.1 Диаграмма вариантов использования	13
2.2 Диаграмма последовательности	15
2.3 Диаграмма деятельности	16
2.4 Диаграмма сущность-связь	17
2.5 Словарь данных	19
3 Разработка интерфейса приложения	20
3.1 Определение интерфейса информационной системы	20
3.2 Анализ интерфейса существующих систем	21
3.3 Создание руководства по стилю	24
3.4 Разработка макетов	25
3.4.1 Анализ методология Канбан и инструмент Канбан-доска	25
3.4.2 Создание макетов	29
4 Разработка приложения	33
4.1 Описание архитектуры программного обеспечения	33
4.2 Создание серверной части	37
4.3 Создание базы данных	40
4.4 Разработка приложения	41
5 Тестирование приложения	45
5.1 Теоретические сведения	45
5.2 Особенности тестирования в проекте	46
5.3 Отчёт о тестировании	46
5.3.1 Позитивные тест-кейсы	47
5.3.2 Негативные тест-кейсы	48
6 Техника безопасности и охрана труда	49
6.1 Вредоносные факторы на стадии разработки приложения	49
6.2 Требования охраны труда перед началом работы	49
6.3 Требования охраны труда во время работы	50
6.4 Требования охраны труда в аварийных ситуациях	50
Заключение	53
Список используемых источников	55
Приложение А	58

Введение

Современный мир существует во время огромного количества информации и возможностей, что приводит к дефициту времени и сил. День состоит из тысячи задач и тысячи отложенных на завтра, для процветания можно не только выполнять поставленные задачи, но и успевать создавать новые и выполнять их лучше.

В ресторанном бизнесе каждодневно меняется ситуация и среда, к которой приходится адаптироваться руководителю ресторана. Необходимо учитывать огромный объём задач начиная с количества брака, заканчивая психологической обстановкой в коллективе, на решение спектра задач, требуется почти идеальное планирование собственного времени и выполнения поставленных задач. Требуется инструмент, который позволит реализовать время и силы руководителя с максимальной эффективностью.

Тема выпускной квалификационной работы актуальна, так как планирование задач с использованием современных технологий, означает, в первую очередь, повышение конкурентоспособности, за счет ускорения процессов руководства, а также повышение качества выполняемых задач, направленных на увеличение эффективности бизнеса.

Объектом автоматизации является разработка приложения для эффективного решения задач управления и задач тайм-менеджмента.

Предметом дипломного проекта является программа для планирования дня руководителя ресторана.

Целью дипломного проекта является разработка приложения для планирования распорядка для руководителя ресторана с использованием гибридной системы планирования.

Для достижения поставленной цели выпускной квалификационной работы необходимо решить следующие задачи:

- Провести анализ предметной области, для которой создается приложение;

- Спроектировать структуру и описать функциональные требования к программному продукту;
- Представить макет интерфейса приложения;
- Спроектировать и создать базу данных с помощью программного средства Microsoft SQL Server Management Studio;
- Выполнить программную реализацию в среде программирования Microsoft Visual Studio;
- Провести тестирование и отладку разработанного приложения.

Разработанное приложение будет обладать всеми необходимыми компонентами для выполнения любых задач, связанных с планировкой задач, основанных на методологии канбан и удобным графическим интерфейсом.

1 Анализ предметной области

1.1 Описание предметной области

Объектом анализа является расписание руководителя ресторана.

Ресторан – это предприятие, занимающиеся производством, реализацией и организацией потребления кулинарной продукции, с широким ассортиментом блюд сложного приготовления, включая заказные и фирменные. Блюда подаются и съедаются на месте в ресторане, но также предлагают блюда на вынос и доставку еды. Рестораны сильно различаются по внешнему виду и предложениям, включая большое разнообразие кухонь и видов обслуживания, от недорогих ресторанов быстрого питания до семейных ресторанов со средней ценой и дорогих заведений класса люкс.

В западных странах в большинстве ресторанов среднего и высокого класса подают алкогольные напитки. В некоторых ресторанах подают все основные блюда, такие как завтрак, обед и ужин.

Зачастую более крупные предприятия имеют свои рестораны, расположенные на их территории, где услуги питания предоставляются для удобства проживающих и для увеличения потенциального дохода гостиницы.

С точки зрения управления, основные бизнес-процессы ресторана с стороны руководителя:

- Выполнение плана по продажам и операционной прибыли;
- Укомплектованность штата, квалификация сотрудников;
- Поддержание атмосферы заведения (температура, запах, влажность, музыкальный ряд, видео ряд, громкость музыки);
- Соблюдение чистоты, санитарии и гигиены;
- Предоставление гостям высокого уровня сервиса и профессиональных продаж;
- Эффективный маркетинг.;
- Контроль расходов и соответствие нормам центров затрат (себестоимость продуктов, заработная плата, аренда, затраты на маркетинг и продажи, административные и производственные расходы);

- Контроль злоупотреблений (пробивка всей продукции по чекам, контроль кассы и расходов).

Для эффективного управления рестораном руководитель выделяет основные бизнес-процессы, которые перерастают в ежедневные задачи:

- Контроль и анализ выручка за день анализирует причины сделать аналитику изменения процентов продаж проверяет табель и планирует графики выходы от выручки;

- Стоп-лист, предоставляемый шеф-поваром анализ и принятие решения Go-лист, предоставляемый шеф-поваром для внесения высоко маржинальные блюда;

- Смотрит отзывы за вчера, анкеты тайных гостей разбираем системные ошибки мотивирование работников;

- Внести ошибки в чат, для обсуждения;

- Проверка журнала брони столов контроль брони обзвон столов за час до брони;

- Проверка работы администраторов: выборочно санузлов, столов, солонки перечницы, выборочно чистоту зала, стейшены, раздача;

- Планерка с официантами и администратором;

- Работа с персоналом (заявки, объявления, собеседования, адаптация, обучение, аттестация, контроль, мотивация);

- Работа с анализом и планированием маркетинга совместная работа с маркетологом;

- Анализ продаж блюд по наценке и востребованности консультация с шеф-поваром;

- Контроль за работой зала;

- Анализ цен и ассортимента конкурентов;

- Бракераж;

- Подписание списаний продуктов, п/ф, посуды и прочего;

- Работа с поставщиками с закупщиком;

- Работа с управляющим технической службы;
- Работа с другими подразделениями (фин. отдел, управляющий, директор);
- Подписание заявок на хозяйственные товары и бар;
- Работа с гостями.

Для контроля объёмного количества бизнес-процессов и их качества, используется три вида отчетов ежемесячные, -недельные, -дневные:

К отчетам, формируемым раз в месяц, относятся:

- План по продажам и прибыли на следующий месяц и защита бюджета;
- Сравнение план-факта по выручке и операционной прибыли и анализ причин невыполнения или перевыполнения;

- Анализ центров затрат и решения;
- Анализ ДДС – движение денежных средств;
- Инжиниринг меню, решения по меню;
- Анализ цен поставщиков и решения;
- Анализ продаж в текущей ситуации по наценке;
- Анализ ошибок, отзывов и анкет тайных гостей;
- Анализ эффективности маркетинга и маркетинговый план.

Отчет, формируемый по итогу недели состоит из даты недели, выручки в рублях, себестоимости продукции и фонда оплаты труда.

Так же к всему вышеперечисленному необходимо добавить отчет, формируемый в конце каждой смены в виде кассового отчета.

При таком количестве ежедневных и не только задач, необходимо держать их голове. Но возникает риск что-то упустить, что в свою очередь может привести к простоям в работе и лишним издержкам.

Количество бизнес-процесса управления изменяется от ресторана к ресторану, завися от их параметров и уровня автоматизации. Поэтому для разработки было выбрано приложение-планировщик с возможностью гибкой настройки.

1.2 Организационная структура организации

Организационная структура – это модель, совокупность подразделений организации, в рамках которой распределяются управленческие задачи, определяются полномочия и ответственность руководителей и должностных лиц, формирующая иерархию внутри компании. Организационная структура устанавливается исходя из целей деятельности и необходимых для достижения этих целей подразделений, выполняющих функции, составляющие бизнес-процессы организации. Схема организационной структуры изображена на рисунке 1, где руководитель управляет всем персоналом в ресторане.

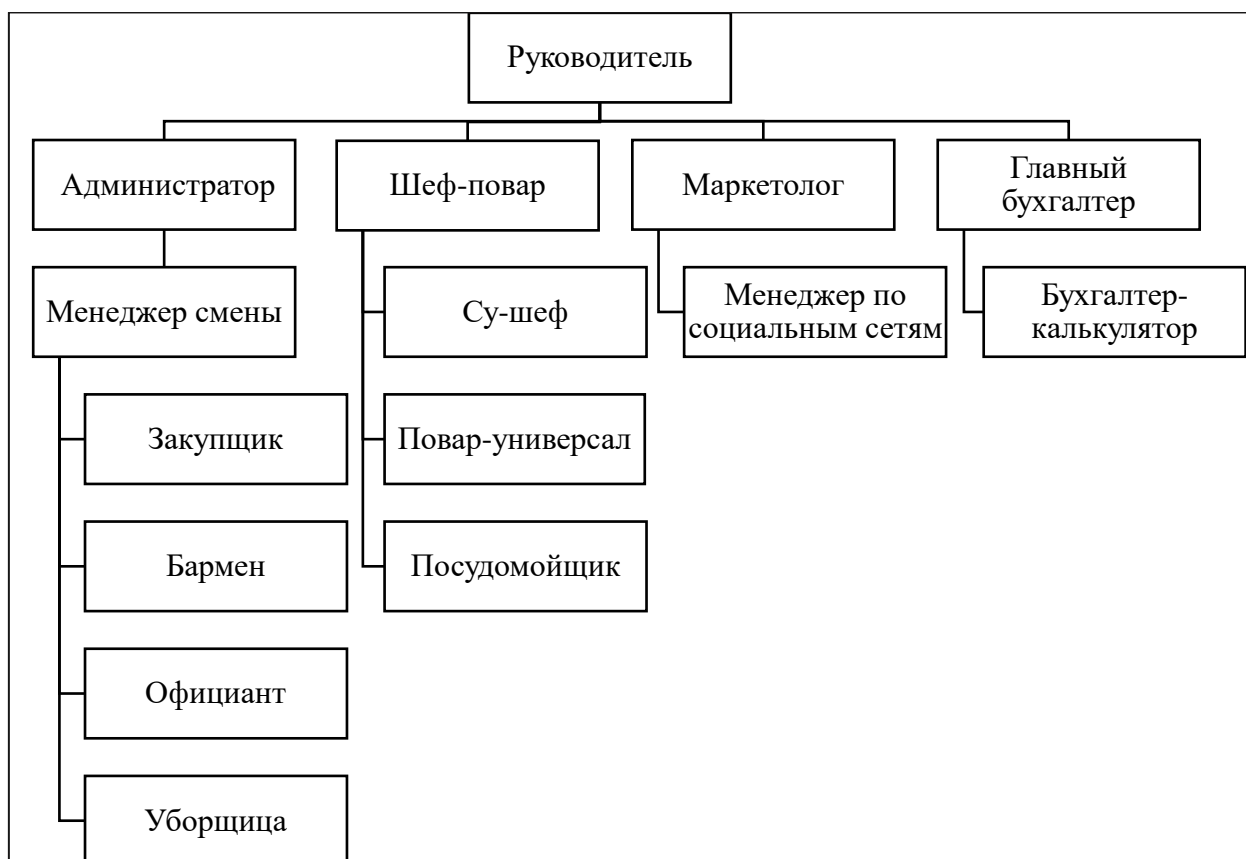


Рисунок 1 – Организационная структура организации

1.3 Контекстная диаграмма

IDEF0 – это методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес-процессов. В роли функциональной модели предназначена для описания существующих бизнес-процессов на предприятии и идеального положения

вещей. Методология IDEF0 предписывает построение иерархической системы диаграмм, единичных описаний фрагментов системы.

Как стандарт IDEF0 представляет организацию как набор модулей, здесь существуют правила сторон:

- стрелка входа всегда приходит в левую кромку активности;
- стрелка управления — в верхнюю кромку;
- стрелка механизма — нижняя кромка;
- стрелка выхода — правая кромка.

Описание выглядит как «чёрный ящик» с входами, выходами, управлением и механизмом, который постепенно детализируется до необходимого уровня.

Построение модели информационной системы начинается с описания функционирования предприятия (системы) в целом в виде контекстной диаграммы. На рисунке 2 представлена контекстная диаграмма информационной системы.

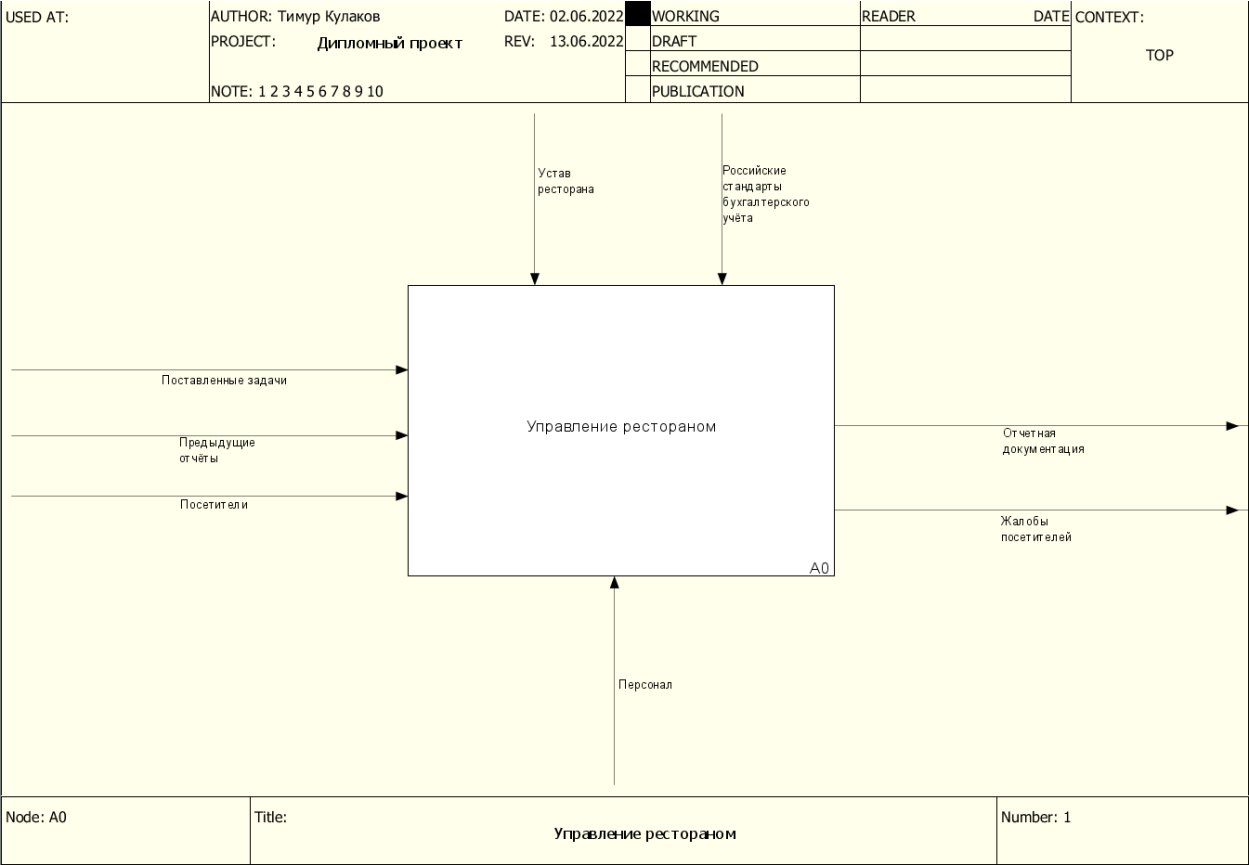


Рисунок 2 – Контекстная диаграмма

При проведении анализа предметной области нужно, прежде всего, определить название основного блока и состав стрелок контекстной диаграммы. В данном проекте рассматривается общая деятельность работы руководителя ресторана, основной задачей которого является управление предприятием, предоставляющего услуги питания и по изготовления продукции общественного питания. Следовательно, основной блок контекстной диаграммы, которому присваивается имя, охватывающее всю сферу деятельности системы, можно назвать – «Управление рестораном». После чего определяем внешние интерфейсы.

Входными объектами данной системы являются поставленные задачи, предыдущие отчеты и посетители.

Выходными объектами системы являются Отчетная документация и жалобы посетителей.

1.4 Диаграмма декомпозиции

Блок A0 – «Управление рестораном», представленный на контекстной диаграмме, может быть описан более подробно с помощью другой диаграммы, расположенной на один уровень ниже в иерархии. Диаграмма нижнего уровня, или диаграмма-потомок, показывает внутреннее содержание блока – родителя. Процесс создания более детальных диаграмм называется декомпозицией.

Декомпозиция является основным понятием стандарта IDEF0, она позволяет постепенно и структурированно представлять модель системы в виде иерархической структуры отдельных диаграмм, что делает ее менее перегруженной и легко усваиваемой. Принцип декомпозиции применяется при разбиении сложного процесса на составляющие его функции. При этом уровень детализации процесса определяется непосредственно разработчиком модели.

На рисунке 3 представлена декомпозиция контекстной диаграммы информационной системы.

2 Проектирование информационной подсистемы

Проектирование информационной системы – это процесс преобразования входной информации об объекте проектирования, а также о методах проектирования и опыте проектирования объектов аналогичного назначения в проект информационной системы в соответствии с имеющимися нормативными требованиями и стандартами.

Необходимо также отметить, что неотъемлемой частью дипломного проекта по проектированию информационной подсистемы является проведение объектно-ориентированного проектирования посредством нотации UML. Язык UML представляет собой графический язык моделирования общего назначения, предназначенный для спецификации, визуализации, проектирования и документирования всех артефактов, создаваемых при разработке программных систем.

При проектировании информационной подсистемы следует сформировать следующие UML диаграммы:

- Диаграмма вариантов использования;
- Диаграмма последовательности;
- Диаграмма деятельности;
- Диаграмма сущность-связь.

Помимо составления диаграмм необходимо, в дополнение диаграмме сущность-связь, составить словарь данных, который будет отображать информацию о данных и их формат.

2.1 Диаграмма вариантов использования

Функциональные требования к ИС описываются с использованием диаграммы прецедентов (вариантов использования). Это диаграмма, предназначена для описаний отношений между actor и прецедентами, и является составной частью модели прецедентов, позволяющей описать функциональные требования к системе.

Диаграмма вариантов прецедентов – это тип поведенческой диаграммы UML, который часто используется для анализа различных систем. Они

позволяют визуализировать различные типы ролей в системе и то, как эти роли взаимодействуют с системой.

Прецедент – возможность моделируемой системы (часть её функциональности), благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат. Прецедент соответствует отдельному сервису системы, определяет один из вариантов её использования и описывает типичный способ взаимодействия пользователя с системой.

Актер – это любая сущность, которая выполняет роль в одной данной системе. Это может быть человек, организация или внешняя система и обычно рисуется как скелет.

Отношения между пользователем системы посредством прецедентов (вариантов использования) представлено в диаграмме прецедентов на рисунке 4.



Рисунок 4 - Диаграмма прецедентов

2.2 Диаграмма последовательности

Диаграмма последовательности — UML-диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл объекта и взаимодействие актеров информационной системы в рамках прецедента.

Основными элементами диаграммы последовательности являются обозначения объектов, вертикальные «линии жизни», отображающие течение времени, прямоугольники, отражающие деятельность объекта или исполнение им определенной функции, и стрелки, показывающие обмен сигналами или сообщениями между объектами.

Эти диаграммы используются для иллюстрации взаимодействия между частями системы и изображают участвующие во взаимодействии объекты и последовательность сообщений, которыми они обмениваются. Диаграммы последовательности применяются тогда, когда требуется посмотреть на поведение нескольких объектов в рамках одного прецедента. На рисунке 5 предоставлена диаграмма последовательности.

Описание сценария данных действий включает следующие действия:

- Авторизацию
- Изменение данных
- Выход из системы

Объекты, используемые в диаграмме:

- Руководитель
- Приложение
- База данных

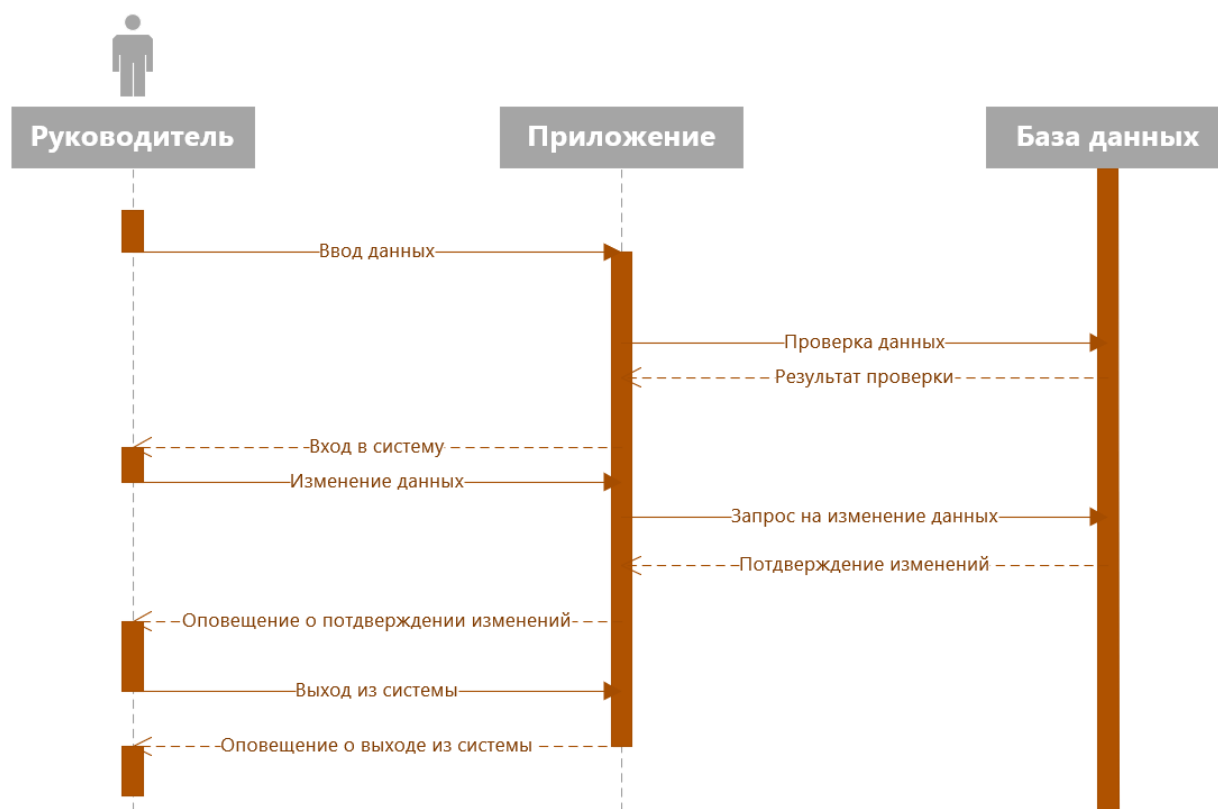


Рисунок 5 – Диаграмма последовательности

2.3 Диаграмма деятельности

Диаграмма деятельности — UML-диаграмма, на которой показаны действия, состояния которых описаны на диаграмме состояний. Под деятельностью понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчинённых элементов — вложенных видов деятельности и отдельных действий, соединённых между собой потоками, которые идут от выходов одного узла ко входам другого.

Диаграммы деятельности используются при моделировании бизнес-процессов, технологических процессов, последовательных и параллельных вычислений.

Диаграммы деятельности состоят из ограниченного количества фигур, соединённых стрелками. Основные фигуры:

Прямоугольники с закруглениями — действия. Узел управления — это абстрактный узел действия, которое координирует потоки действий.

Ромбы — решения. Узел решения предназначен для определения правила ветвления и различных вариантов дальнейшего развития сценария. В точку ветвления входит ровно один переход, а выходит — два или более.

Широкие полосы — начало и окончание ветвления действий. Узел объединения имеет два и более входящих узла и один исходящий.

Чёрный круг — начало процесса. Начальный узел деятельности является узлом управления, в котором начинается поток при вызове данной деятельности извне.

Чёрный круг с обводкой — окончание процесса. Конечный узел деятельности является узлом управления, который останавливает все потоки данной диаграммы деятельности. На диаграмме может быть более одного конечного узла.

Стрелки идут от начала к концу процесса и показывают потоки управления или потоки объектов.

Диаграмма деятельности для прецедента «Создать задачу в списке задач состояния» представлена в Приложении А.

2.4 Диаграмма сущность-связь

Схема «сущность-связь» — это разновидность блок-схемы, где показано, как разные «сущности» связаны между собой внутри системы. ER-диаграммы чаще всего применяются для проектирования и отладки реляционных баз данных. ER-диаграммы полагаются на стандартный набор символов, включая прямоугольники, ромбы, овалы и соединительные линии, для отображения сущностей, их атрибутов и связей.

Моделирование данных проводится для того, чтобы разработчики информационной системы получили в свое распоряжение модель (или модели), описывающую систему баз данных. Для моделирования данных чаще всего используется диаграмма «сущность-связь», при помощи которой выделяются важные сущности, их свойства и отношения друг с другом.

С помощью ERD определяются важные для конкретного случая объекты или сущности, их отношения, свойства и атрибуты. ER-диаграммы

применяются для моделирования и проектирования реляционных баз данных, которые помогают оптимизировать процессы, извлекать данные и повышать качество результатов.

В сфере разработки программного обеспечения ER-диаграмма служит первым шагом в определении требований проекта по созданию информационных систем. ER-диаграммы применяются для анализа уже имеющихся баз данных с целью выявить и устранить ошибки в логике или развертывании. Диаграмма позволяет выявить, где именно закрались ошибки.

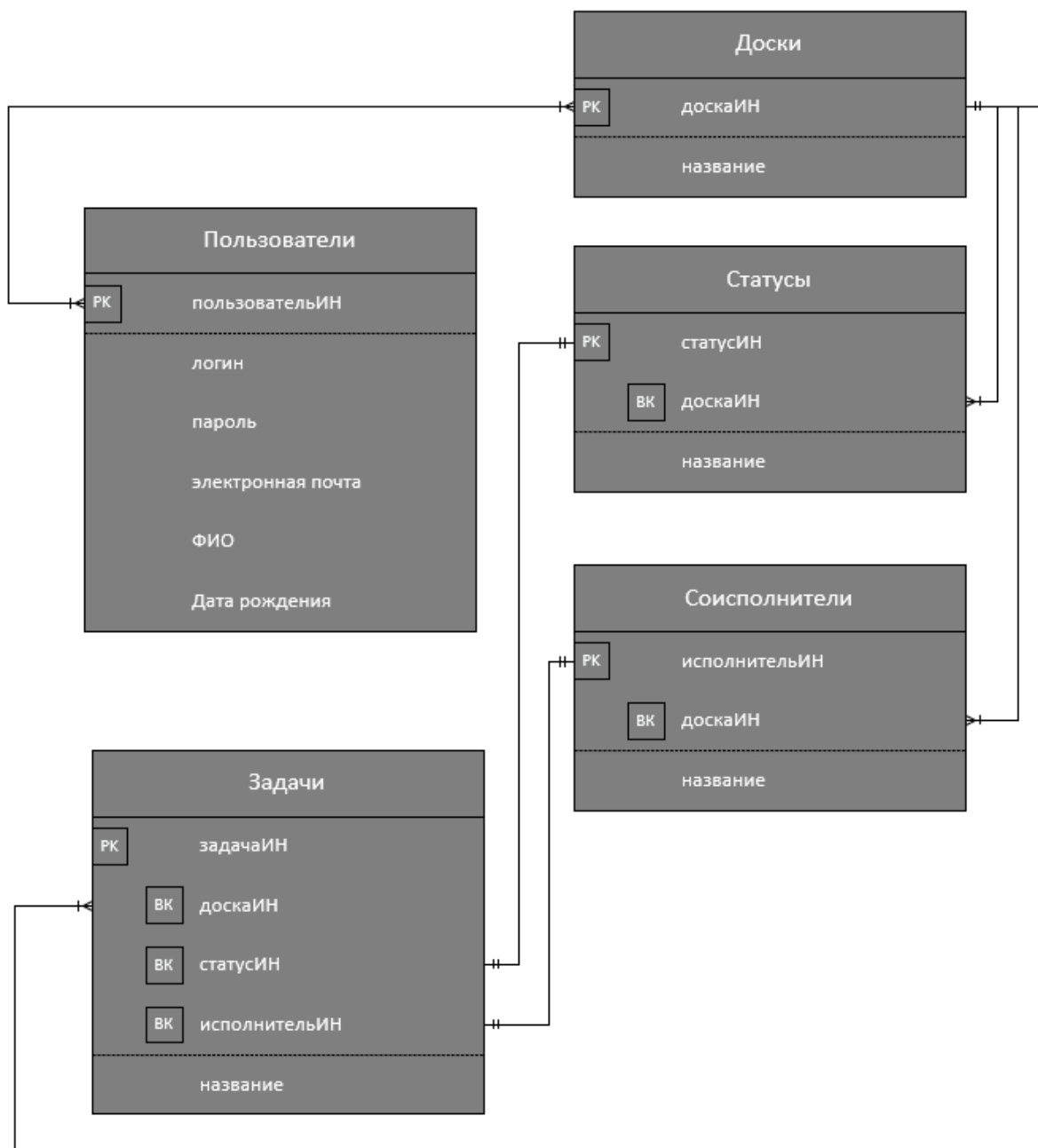


Рисунок 6 – Диаграмма сущность-связь

2.5 Словарь данных

Для разрабатываемой базы данных приложения был составлен словарь данных, который содержит наименование столбца, тип хранимых в нём данных, является ли оно обязательным, а также информацию о том, является ли этот столбец первичным или внешним ключом. Словарь данных представлен на таблицах 1-6.

Таблица 1 – Таблица «Статусы»

Ключ	Наименование	Тип данных	Обязательное
Первичный ключ	statusID	int	+
Внешний ключ	kanbanID	int	+
	name	nvarchar(50)	+

Таблица 2 – Таблица «Задачи»

Ключ	Наименование	Тип данных	Обязательное
Первичный ключ	taskID	int	+
Внешний ключ	kanbanID	int	+
Внешний ключ	statusID	int	+
Внешний ключ	toolID	int	+
	name	nvarchar(50)	+

Таблица 3 – Таблица «Соисполнители»

Ключ	Наименование	Тип данных	Обязательное
Первичный ключ	toolID	int	+
Внешний ключ	kanbanID	int	+
	name	nvarchar(50)	+

Таблица 4 – Таблица «Канбаны»

Ключ	Наименование	Тип данных	Обязательное
Первичный ключ	kanbanID	int	+
	name	nvarchar(50)	+

Таблица 5 – Таблица «Пользователи»

Ключ	Наименование	Тип данных	Обязательное
Первичный ключ	userID	int	+
	login	nvarchar(50)	+
	password	nvarchar(50)	+
	email	nvarchar(50)	+
	fullname	nvarchar(50)	+
	birthday	date	+
	joindate	date	+

Таблица 6 – Таблица «Пользователи и Канбаны»

Ключ	Наименование	Тип данных	Обязательное
Первичный ключ	userID	int	+
Первичный ключ	kanbanID	int	+

3 Разработка интерфейса приложения

3.1 Определение интерфейса информационной системы

Пользовательский интерфейс представляет собой совокупность программных и аппаратных средств, осуществляющих взаимодействие пользователя с информационной системой. Пользовательский интерфейс включает в себя систему меню, диалоговые формы, сообщения об ошибках, справочную систему и т.п.

Пользовательский интерфейс – это пространство, в котором происходит взаимодействие между пользователем и системой. Цель этого взаимодействия обеспечить эффективную эксплуатацию и управление системой со стороны пользователя, в то время как система передает информацию, которая помогает пользователю в процессе принятия решений.

Как правило, целью проектирования пользовательского интерфейса является создание интерфейса, который позволяет легко, эффективно и удобно управлять системой таким образом, чтобы получить желаемый результат.

Это означает, что пользователю необходимо обеспечить минимальные входные данные для достижения желаемого результата, а также что система сводит к минимуму нежелательные выходные данные.

Пользователь информационной системы взаимодействует с интерфейсом: через интерфейс посылает входные данные и принимает выходные.

Процессы по выполнению заданий вызываются интерфейсом в требуемые моменты времени. Поэтому интерфейс — это основной процесс, а процессы, выполняющие различные задания, являются неосновными, или фоновыми.

Условно интерфейсы ИС можно разделить на 3 группы:

- Текстовые (текст-ориентированные);
- Смешанные (псевдографические);
- Графические.

Для использования текстового интерфейса необходимо хорошо знать операционную систему, а сам процесс общения с системой требует от пользователя высокой квалификации и состоит во введении в нужной последовательности большого числа трудно запоминаемых команд в командную строку.

Псевдографический интерфейс можно отнести к промежуточному между чисто командным интерфейсом и графическим. Он в большинстве случаев обладает преимуществами первого, использование мощных языков, расширяемость, и устраняет некоторые недостатки, позволяет легче управлять системой, нагляднее представлять структуру системы.

Графический интерфейс пользователя заключается в использовании представления информации с помощью графических элементов. Графический интерфейс получил неофициальную характеристику — «интуитивно понятный».

3.2 Анализ интерфейса существующих систем

При анализе интерфейса существующих систем необходимо учитывать, какие решения они предлагают, программное обеспечение или библиотеку для внутренней разработки.

Из систем с готовым программным обеспечением выделяются:

- Kaiten – российская компания, которая разработала одноименную систему совместного управления процессами и командами. На рисунке 6 предоставлен интерфейс.

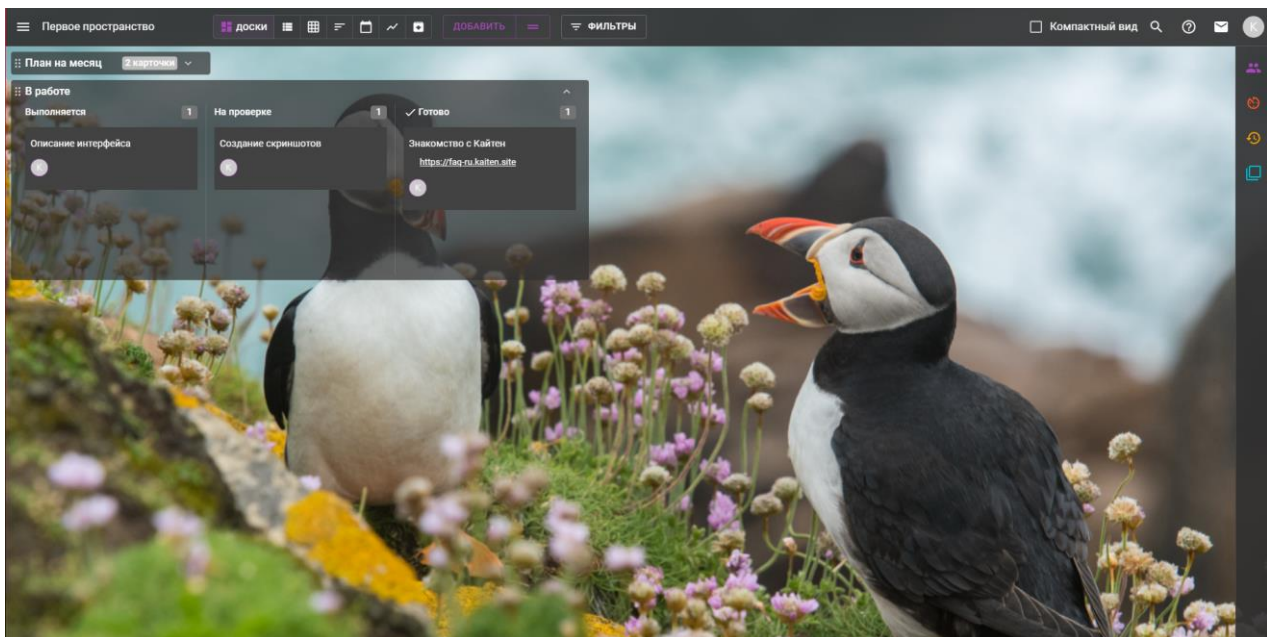


Рисунок 7 – Интерфейс Kaiten

- Trello – облачная программа для управления проектами небольших групп, разработанная Fog Creek Software. На рисунке 7 предоставлен интерфейс.

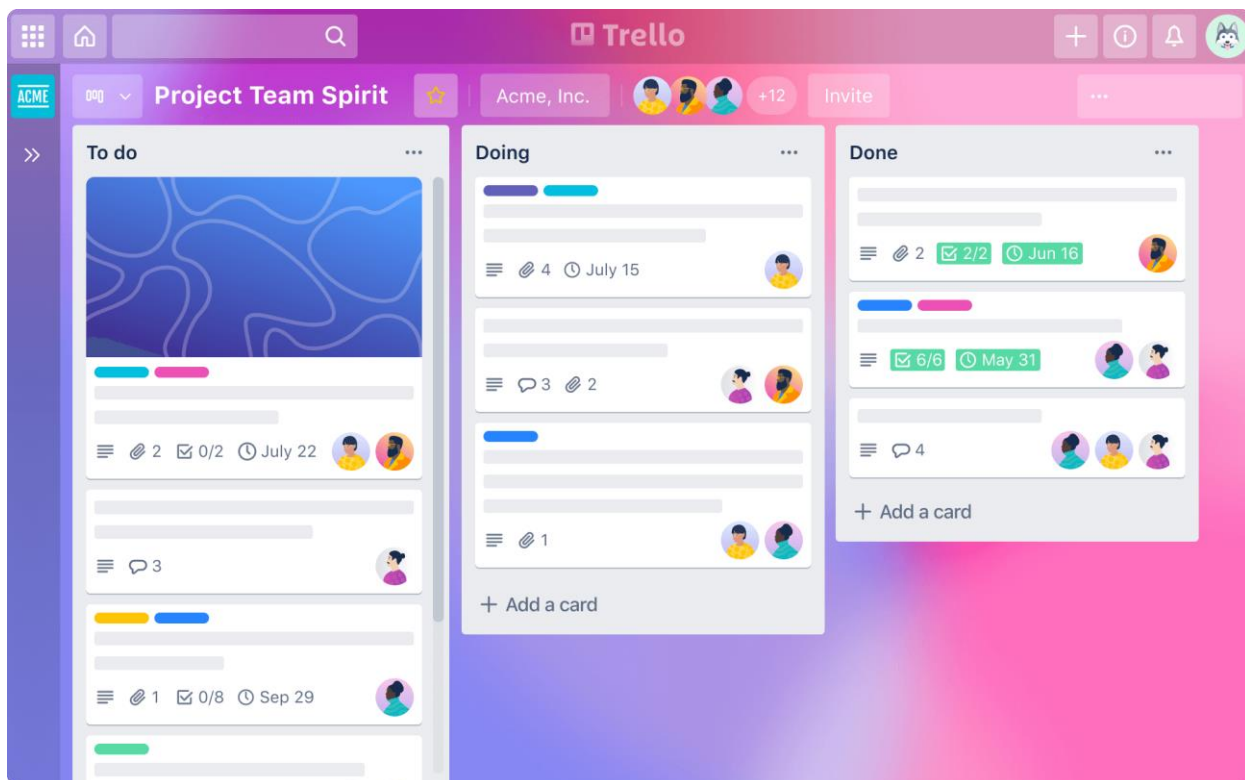


Рисунок 8 – Интерфейс Trello

- Jira – это программный инструмент для управления проектами, разработанный компанией Atlassian. На рисунке 8 предоставлен интерфейс.

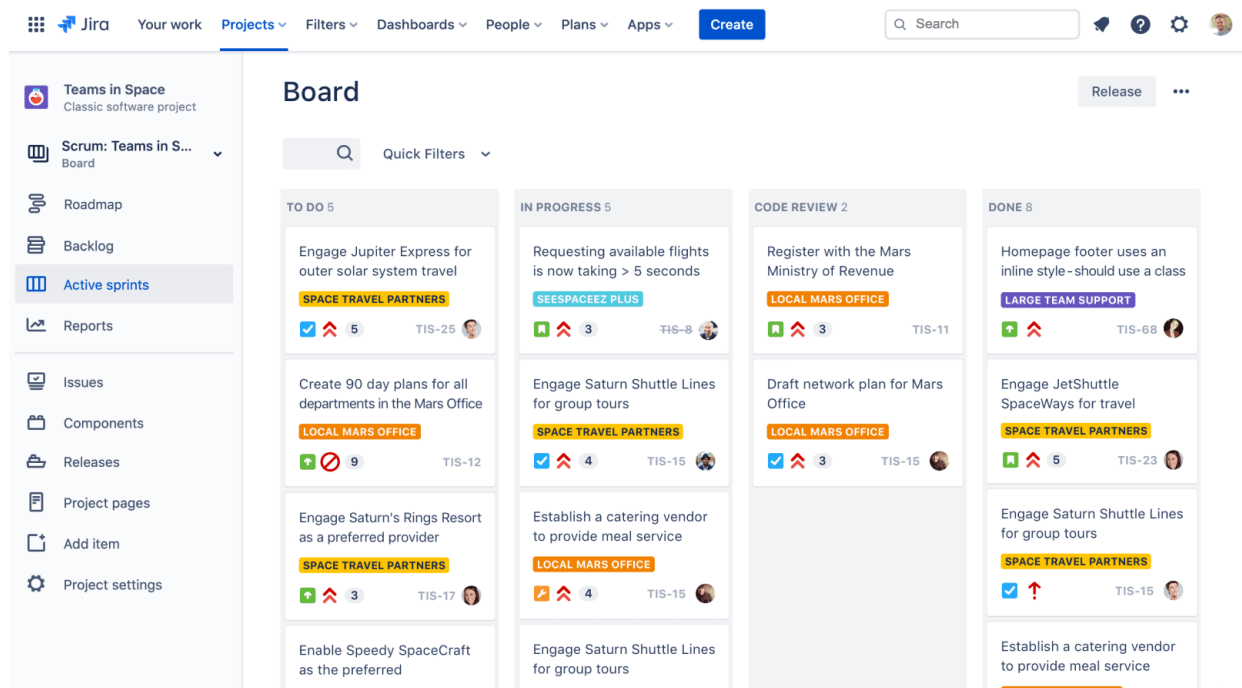


Рисунок 9 – Интерфейс Jira

Среди конкурентов с библиотекой для разработки на языке C#, главенствует компания DevExpress занимается разработкой компонентов и дополнений для WinForms, ASP.NET, WPF. Библиотека обладает большой функциональностью, удобством и простотой использования. Компания предоставляет высококачественную профессиональную техподдержку клиентов и имеет статус Microsoft Gold Certified Partner. На рисунке 9 предоставлен интерфейс.

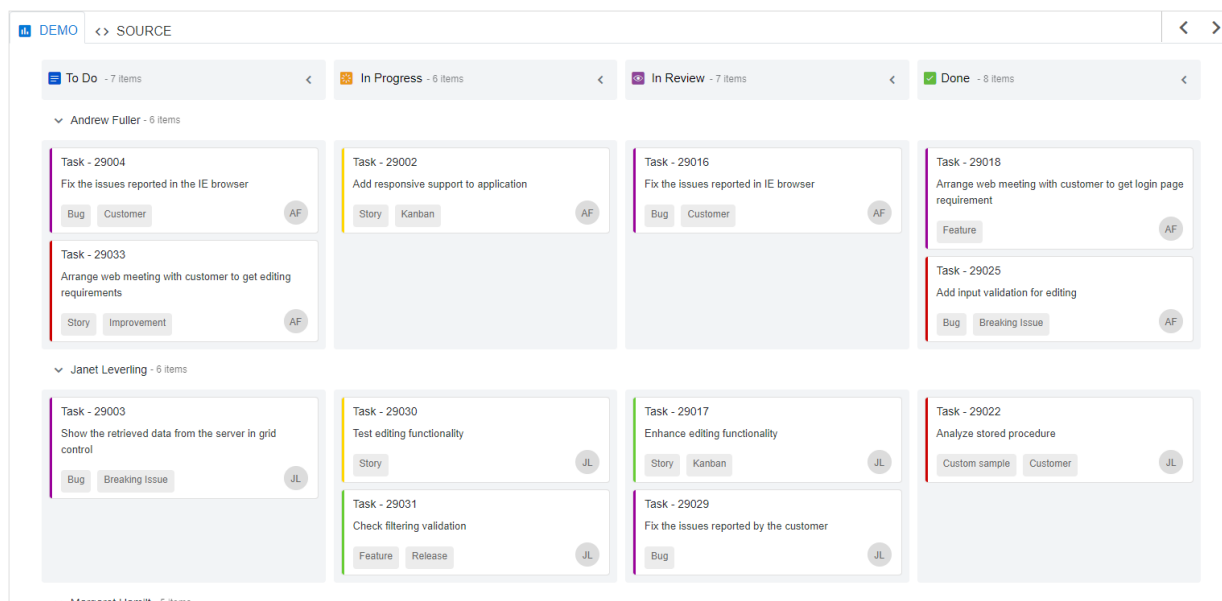


Рисунок 10 – Интерфейс, разработанной на библиотеке DevExpress

В итоге анализа интерфейсов конкурентов было установлено, что при создании приложений планирования используют, определенные разработчиками, методологии и их инструменты, используют моногамную цветовую палитру и плоский дизайн элементов управления интерфейсом.

3.3 Создание руководства по стилю

Руководство по стилю — это набор стандартов, которые подстраивают дизайн под миссию и тон компании.

На макетах пользовательского интерфейса было использованно 3 основных цвета под цвет компании, предоставлены на рисунке 11 в виде цветовой палитры:

- #cee5ec (206,229,236)
- #ffffff (255,255,255)
- #f0f0f0 (240,240,240)

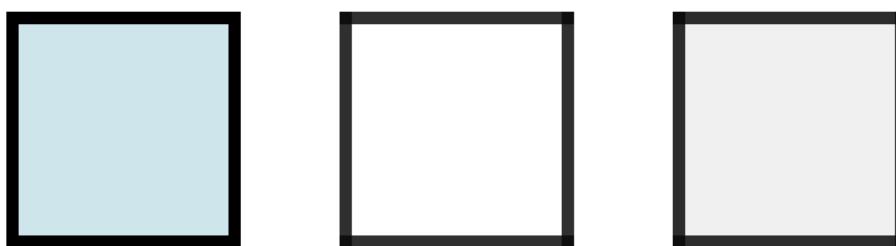


Рисунок 11 – Палитра используемых цветов

На каждой форме будет отображаться логотип (рисунок 12) приложения, для более каноничного сходства его расположение будет наверху форм в левом углу.



Рисунок 12 – Логотип

Так же для точного и удобного пользовательского интерфейса будет использован шрифт в читаемом и понятном виде:

Шрифт для всех названий форм и кнопок Segoe UI Historic.

Шрифт для названия форм – полужирный;

Шрифт для названия кнопок – обычный.

Для повышения презентабельности приложения будет использоваться open-source набор бесплатных иконок для пользовательского интерфейса – Core Line – Free, по лицензии Creative Commons - CC BY 4.0. Основан он на 14 пиксельной сетки и векторной графике с регулируемой толщиной линий.

3.4 Разработка макетов

Макет интерфейса – это статическая схематическая иллюстрация формы, на которой отображено расположение и содержание отдельных элементов интерфейса пользователя. Задача макета – показать содержание конкретной страницы, взаимное расположение блоков и отдельных элементов. В абсолютном большинстве случаев макеты интерфейса не раскрывают особенностей визуального дизайна.

3.4.1 Анализ методология Канбан и инструмент Канбан-доска

Канбан — это метод, демонстрирующий, что происходит в процессе работы. Канбан-метод базируется на визуализации нематериальной деятельности, благодаря методу у нас формируется понимание того, какую работу мы выполняем, по каким правилам, с каким объемом задач можем справиться за единицу времени и какой результат мы предоставляем.

По достижении такого понимания можно переходить к совершенствованию задач и процессов. На этом этапе увеличивается предсказуемость процедур, работа становится более равномерной. Усиливается взаимодействие, а вместе с ним повышается качество. Сотрудники все чаще ведут работу самостоятельно, поскольку начинают разбираться в управлении рисками.

Канбан также можно использовать, чтобы достичь большей согласованности действий в рамках предприятия, а это означает быстрое достижение стратегических целей.

Канбан акцентирует внимание на управлении обязательствами и равномерном распределении работы, что приводит к повышению гибкости организации. Если изменятся рыночные условия или возникнут проблемы, связанные с управлением зависимостями, Канбан обеспечит оперативную смену курса. Именно поэтому Канбан представляет собой альтернативный путь к бизнес-гибкости.

Основные практики Канбана определяют базовые действия лиц, занимающихся управлением канбан-системами. Всего существует шесть таких практик:

1. Визуализируй;
2. Ограничивай количество незавершенной работы;
3. Управляй потоком;
4. Делай правила работы явными;
5. Внедряй циклы обратной связи;
6. Улучшайтесь совместно, эволюционируйте на основе экспериментов.

Все перечисленные практики вовлекают в:

- способность видеть, в чем заключается рабочий процесс;
- совершенствование процессов, т. е. сохранение и распространение полезных изменений, а также изучение, корректировка и сдерживание неэффективных изменений.

Для конечного утверждения понятия методологии канбан и её выбора в качестве реализации гибкого подхода к планированию, стоит определить её отличия от методологии SCRUM, с которой её чаще всего противопоставляют.

Scrum — это готовое руководство о том, как организовать итеративно-инкрементальную разработку нового продукта. Есть даже инструкция, которая называется Scrum Guide. Все элементы Scrum взаимосвязаны друг с

другом, и при реализации Scrum нельзя выбросить ничего из того, что указано в Scrum Guide.

Scrum — это не инструмент менеджмента, а способ организации рабочих процессов для разработки новых продуктов.

Scrum изначально создавался, чтобы быстро разработать инновационный продукт, подобных которому может еще даже в мире не существует. Он может быть реализован в отдельных продуктовых подразделениях и не требует обязательного изменения рабочих процессов других подразделений или в компании в целом.

Встречи в Scrum рассчитаны на активное вовлечение всех участников команды и коллективное принятие решений.

Kanban-метод ставит своей конечной целью дать возможность целым компаниям быстро реагировать на изменения рынка. Канбан-метод делает это за счет такого построения рабочих процессов, которое позволяет быстро перестраиваться их под новые реалии. Чтобы этого достичь требуется использовать целостный взгляд на компанию для того, чтобы увидеть, где именно кроются препятствия, не позволяющие работать быстро и гибко.

Kanban-метод — это инструмент менеджмента, то есть он предназначен для руководителей.

Канбан-метод существует по принципу ящик с инструментами. Можно использовать только что-то одно, или все сразу. Каждый инструмент приносит свою пользу. Выбор инструмента зависит лишь от готовности его применять. Есть разные степени зрелости применения Канбан-метода, и на каждом уровне организации использует те или иные его элементы и их сочетания.

Цель встреч заключается в том, чтобы проанализировать собранные данные о рабочем процессе, выделить системные проблемы и принять управленческие решения способами изменения рабочего процесса ради устранения выявленных проблем.

Инструментальная разница между Канбан и SCRUM:

- В Канбан нет таймбоксов ни на задачи, ни на спринты;

- В Канбан задачи больше и их меньше;
- В Канбан оценки сроков на задачу опциональные;
- В Канбан «скорость работы команды» отсутствует и считается только среднее время на полную реализацию задачи;

Различия Scrum и Канбан-метод, можно завершить, количественной затратой на внедрение. При Scrum необходимо обучить всю команду разработки и менеджмента для эффективной работы с методологией. В свою очередь Канбан-метод может быть внедрен при обучении всего одного человека из состава менеджмента.

Фреймворк Scrum и Канбан-метод строятся на совершенно разных принципах, фокусируются на разных аспектах рабочей деятельности и предназначены для разных целей. Но при этом, инструменты Канбан-метода можно применять при работе по Scrum.

Доска канбана — это один из инструментов, который можно использовать для внедрения канбана для управления работой на личном или организационном уровне.

Доски Канбан визуально отображают работу на различных этапах процесса, используя карточки для представления рабочих элементов и столбцы для представления каждого этапа процесса. Карточки перемещаются слева направо, чтобы показать прогресс и помочь координировать команды, выполняющие работу. Доска канбан может быть разделена на горизонтальные "плавательные полосы", представляющие различные виды работ или разные команды, выполняющие работу.

Доски Канбан можно использовать в умственной работе или для производственных процессов.

Простая доска имеет столбцы для "ожидание", "в процессе" и "завершено" или "задачи", "выполнение" и "готово". Можно создавать сложные доски канбана, которые разделяют "незавершенную" работу на несколько столбцов, чтобы визуализировать поток работ по всей карте потока создания ценности.

Основные принципы доски канбан:

- визуализация рабочего процесса;
- ограничение работы, которая находится в процессе;
- перемещение задач от колонки к колонке;
- мониторинг, адаптация и оптимизация.

В итоге анализа методологии канбан и её инструмента доски канбан, приложение будет разработано под принципы доски канбан, для повышения эффективности приложения, за счет использования современных решений.

3.4.2 Создание макетов

При входе в приложение первое, что предстает перед пользователем это окно Авторизации, показано на рисунке 13, на котором можно ввести логин и пароль для входа в систему.



Авторизация

Лого

Авторизация

логин

пароль

Вход

Регистрация

Рисунок 13 – Форма Авторизации

Если у пользователя отсутствует зарегистрированный аккаунт, он может нажать кнопку «Регистрация», после нажатия откроется форма регистрации, показана на рисунке 14.

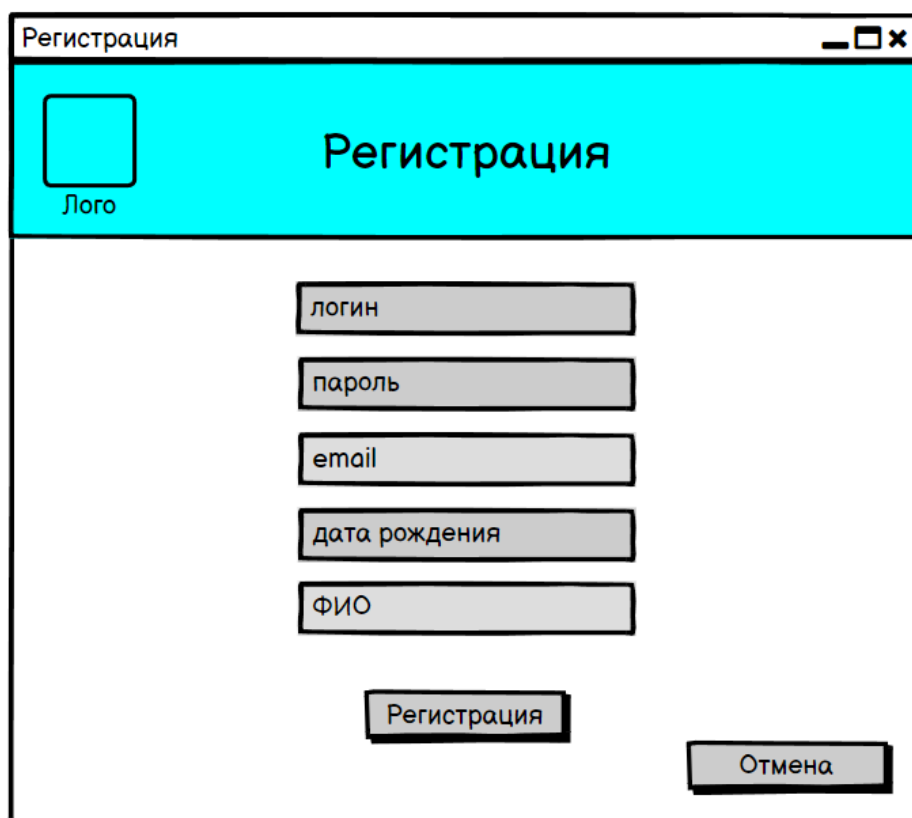


Рисунок 14 – Форма Регистрации

После удачной авторизации, перед нами открывается форма Канбан, представлена на рисунке 15, в этой форме происходят основные действия приложения, такие как выбор канбана, его создание и изменение, взаимодействие с статусами и задачами, их удаление, изменение и создание.

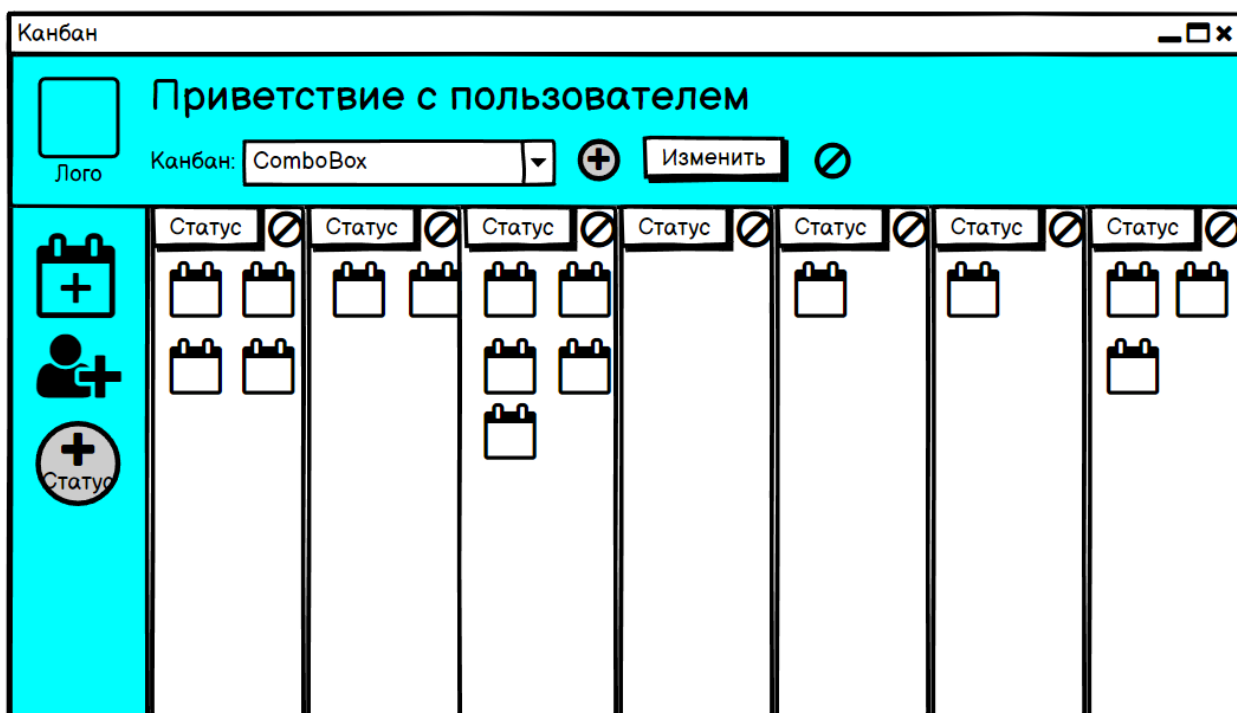


Рисунок 15 – Форма Канбан

Взаимодействие с задачами происходит посредством формы Изменение\Создание задачи, представлена на рисунке 16, открываемой двойным кликом по задаче.



Изменение\Создание задачи

Название

Канбан

Исполнитель

Статус

Сохранить

Отмена

Рисунок 16 – Форма редактирования задач

Соисполнители

Лого

Соисполнители

Таблица соисполнителей

Имя

Канбан

Добавить

Изменить

Удалить

Назад

Рисунок 17 – Форма Соисполнители

Из формы Канбан можно обратиться к форме Соисполнители, предоставлена на рисунке 17, в которой можно взаимодействовать с соисполнителями задач, удалять, изменять и создавать.

4 Разработка приложения

4.1 Описание архитектуры программного обеспечения

Программное обеспечение разрабатываемого приложения было реализовано в виде клиент-серверной архитектуры, схема которой представлена на рисунке 18.

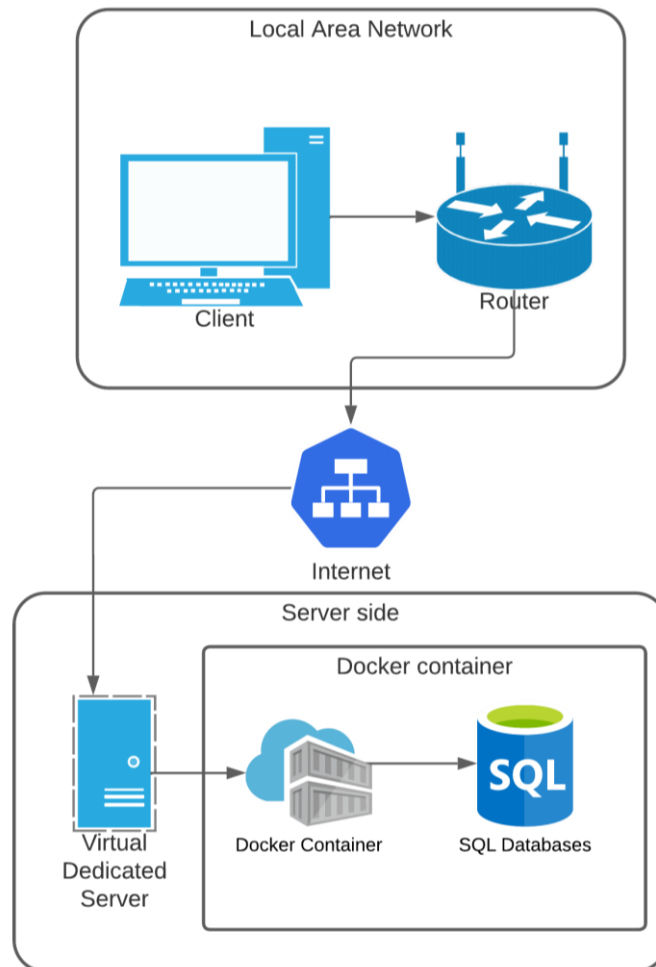


Рисунок 18 – Схема архитектуры программного обеспечения «KanbanAPP»

В качестве системы управления базами данных используется Microsoft SQL Server. Microsoft SQL Server – это система управления реляционными базами данных (РСУБД). Данное СУБД позволяет хранить большие объемы данных и выполнять различные манипуляции с ними (сохранение, выборка, изменение и удаление). Она может использоваться для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия.

Виртуальный выделенный сервер запускает собственную копию операционной системы, и клиенты могут иметь доступ на уровне

суперпользователя к этому экземпляру операционной системы, поэтому они могут устанавливать практически любое программное обеспечение, работающее на этой ОС. Для многих целей он функционально эквивалентен выделенному физическому серверу и, будучи программно-определяемым, может быть создан и настроен гораздо проще. Виртуальный сервер стоит намного дешевле, чем эквивалентный физический сервер. Однако, поскольку виртуальные серверы совместно используют базовое физическое оборудование с другими VPN, производительность может быть ниже в зависимости от рабочей нагрузки любых других выполняющихся виртуальных машин.

Для развертывания SQL Server был арендован виртуальный выделенный сервер на платформе firstvds.ru, интерфейс развертывания представлен на рисунке 19.

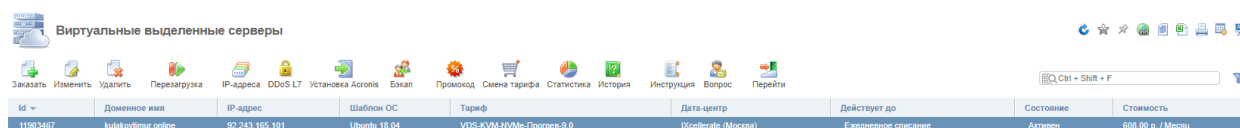


Рисунок 19 – интерфейс firstvds для аренды виртуальных выделенных серверов.

Технические характеристики арендованного сервера:

- Тарифный план: VDS-KVM-NVMe-Прогрев-9.0
- Параметры: 1 ядро процессора, 2 Гб оперативной памяти, 20 Гб диска, 1 IP-адрес

- Запущен: 2022-05-02
- Доменное имя: kulakovtimur.online
- Операционная система: Ubuntu 18.04

Управление арендованным сервером осуществлялось при помощи операционной системы Ubuntu 18.04 и ISPmanager — панель управления сервером.

Ubuntu - дистрибутив Linux, основанный на Debian и состоящий в основном из свободного программного обеспечения с открытым исходным кодом. Ubuntu официально выпущена в трех версиях: настольной, Серверной, и основной для смарт-устройств и роботов. Все выпуски могут запускаться как

на одном компьютере, так и на виртуальной машине. Ubuntu - популярная операционная система для облачных вычислений с поддержкой OpenStack. Рабочим столом Ubuntu по умолчанию является GNOME начиная с версии 17.10.

Ubuntu выпускается каждые шесть месяцев, а выпуски долгосрочной поддержки (LTS) - каждые два года. По состоянию на 21 апреля 2022 года самым последним выпуском долгосрочной поддержки является 22.04.

Ubuntu разрабатывается британской компанией Canonical и сообществом других разработчиков в рамках меритократической модели управления. Canonical предоставляет обновления безопасности и поддержку для каждого выпуска Ubuntu, начиная с даты выпуска и до тех пор, пока выпуск не достигнет установленного срока годности (EOL) дата. Canonical получает доход за счет продажи премиум-сервисов, связанных с Ubuntu, и пожертвований от тех, кто скачивает программное обеспечение Ubuntu.

Ubuntu названа в честь философии ubuntu Нгуни, что, как указывает Canonical, означает "человечность по отношению к другим" с оттенком "Я такой, какой я есть, потому что мы все такие".

Debian GNU/Linux – операционная система, состоящая из свободного ПО с открытым исходным кодом. В настоящее время Debian – один из самых популярных и важных дистрибутивов GNU/Linux, в первичной форме оказавший значительное влияние на развитие этого типа ОС в целом. Debian является универсальной операционной системой и может использоваться в качестве операционной системы как для серверов, так и для рабочих станций.

На сервере был запущен Docker-контейнер с образом MS SQL Server 2019.

Docker – программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации, контейнеризатор приложений. Позволяет «упаковать» приложение со всем его окружением и зависимостями в контейнер, который может быть развёрнут на

любой Linux-системе с поддержкой контрольных групп в ядре, а также предоставляет набор команд для управления этими контейнерами.

Для конфигурации сервера использовался SSH-клиент PuTTY. Он позволяет подключиться и управлять удаленным узлом (например, сервером). В PuTTY реализована только клиентская сторона соединения — сторона отображения, в то время как сама работа выполняется на стороне сервера.

Помимо конфигурации сервера по SSH, так же использовался программный продукт ISPmanager — коммерческая панель управления веб хостингом позволяет управлять программным обеспечением, таким как веб сервер, сервер баз данных, почтовый сервер и другими сопутствующими программами через веб-интерфейс.

Для выполнения работы с сервером базы данных с использованием графического пользовательского интерфейса применяется программа SQL Server Management Studio (SSMS). Она позволяет выполнять конфигурирование, управление и администрирование всеми компонентами Microsoft SQL Server. SSMS включает редактор SQL-скриптов и графическую программу, которая работает с объектами и настройками сервера. Главным инструментом SSMS является компонент Object Explorer, который позволяет пользователю просматривать, извлекать объекты сервера, а также полностью управлять ими.

Для визуализации графического пользовательского интерфейса используется Windows Forms — интерфейс программирования приложений, отвечающий за графический интерфейс пользователя и являющийся частью Microsoft .NET Framework. Данный интерфейс упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обёртки для существующего Win32 API в управляемом коде.

Клиентская часть программного обеспечения была разработана на языке C# с использованием среды разработки Microsoft Visual Studio 2022.

4.2 Создание серверной части

Для создания серверной части был создан виртуальный сервер и выбрана операционная система. Параметры создания виртуального сервера приведён на рисунке 20:

The screenshot displays the 'VDS Старт' (VDS Start) configuration interface. At the top, it specifies hardware resources: CPU: 1 Ядер (1 Core), ОЗУ: 2 Гб (2 GB RAM), and Диск: NVMe 40 Гб (40 GB NVMe Disk). The order period is set to '1 месяц' (1 month). The quantity is '1' with a total price of '509,00 ₽'. The operating system is 'Ubuntu 18.04'. Pre-installed software includes 'ISPmanager 6 Lite' (selected), 'ISPmanager 6 Pro', 'ISPmanager 6 Host', 'Openvpn', 'Teamspeak', and 'Tomcat'. Backup is disabled. Channel options include 'до 100 Мбит/с - безлимит' (selected), 'до 1 Гбит/с - 32 Тб/мес', and 'Защита от DDoS (1 Мбит/с)'. IPv4 addresses are set to 1. IPv6 subnet is disabled. A sidebar on the right shows the total price of '509,00 ₽' and a green 'Оплатить' (Pay) button.

Рисунок 20 – Параметры создания виртуального сервера

Для подключения к серверу требуется SSH-сервер, статический IP-адрес.

SSH – сетевой протокол прикладного уровня, позволяющий производить удалённое управление операционной системой и туннелирование TCP-соединений (например, для передачи файлов). Схож по функциональности с протоколами Telnet и rlogin, но, в отличие от них, шифрует весь трафик, включая и передаваемые пароли. SSH допускает выбор различных алгоритмов

шифрования. SSH-клиенты и SSH-серверы доступны для большинства сетевых операционных систем.

После этого требуется произвести запуск программы PuTTY, в которой нужно указать IP-адрес целевого сервера. Конфигурация подключения изображена на рисунке 21:

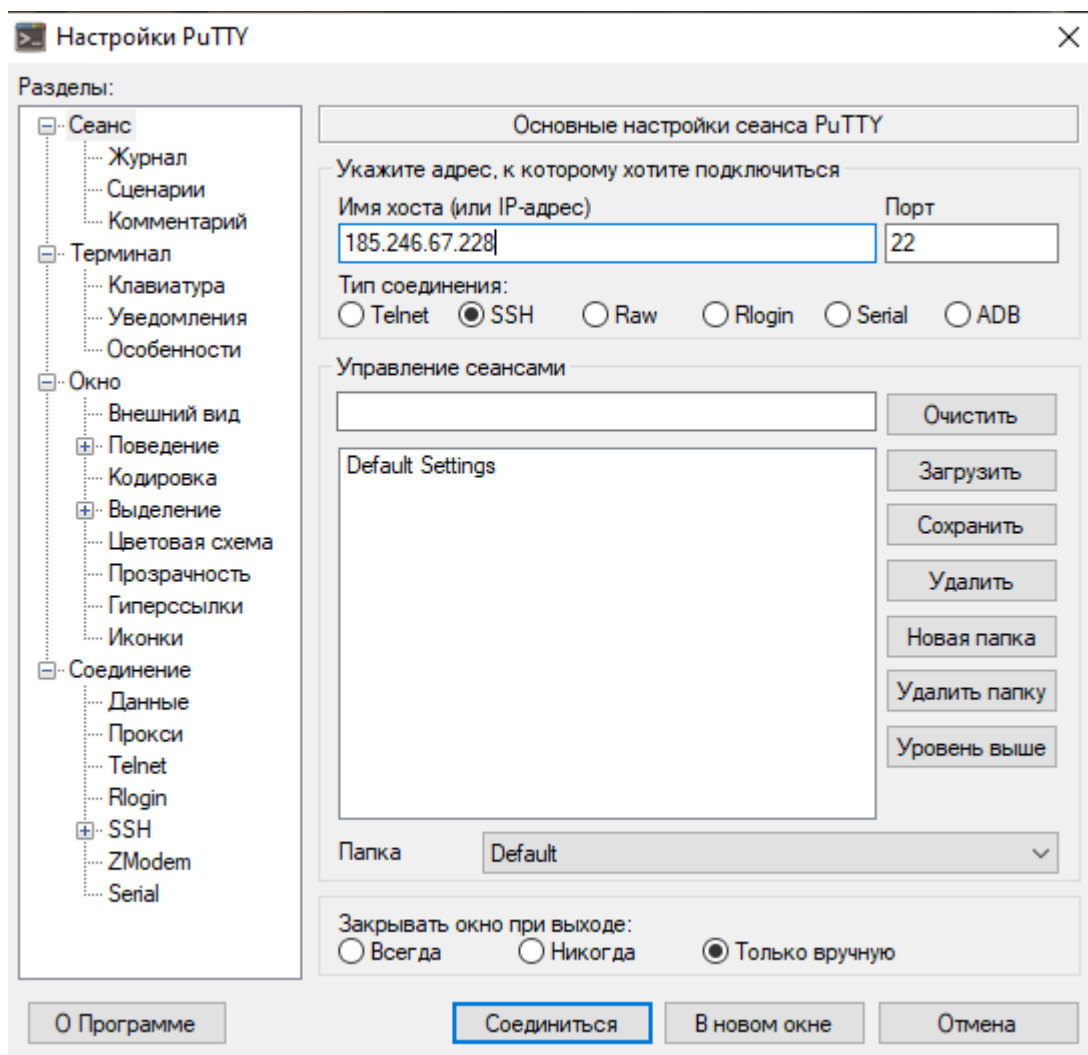


Рисунок 21 – Конфигурация программы PuTTY для подключения к серверу

После первого включения сервера была выполнена команда обновления репозитория:

```
1. sudo apt-get update
```

Результат выполнения команды приведён на рис. 22:

```
root@restkn: ~  
login as: root  
root@185.246.67.228's password:  
welcome to Ubuntu 18.04.6 LTS (GNU/Linux 4.15.0-20-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:       https://ubuntu.com/advantage  
Last login: Thu May  5 10:35:00 2022 from 109.252.177.82  
root@restkn:~# lsb_release -a  
No LSB modules are available.  
Distributor ID: Ubuntu  
Description:   Ubuntu 18.04.6 LTS  
Release:       18.04  
Codename:      bionic  
root@restkn:~# sudo apt-get update  
Hit:1 http://download.ispsystem.com/repo/ubuntu 6-5.346-bionic InRelease  
Hit:2 http://download.ispsystem.com/repo/ubuntu base-bionic InRelease  
Hit:3 http://download.ispsystem.com/repo/ubuntu 5.346-bionic InRelease  
Get:4 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]  
Hit:5 http://us.archive.ubuntu.com/ubuntu bionic InRelease  
Get:6 http://us.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]  
Get:7 http://us.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]  
Fetched 252 kB in 1s (274 kB/s)  
Reading package lists... Done  
root@restkn:~#
```

Рисунок 22 – Результат выполнения команды обновления репозитория

После обновления репозитория был установлен Docker с помощью команд:

1. `sudo apt install apt-transport-https ca-certificates curl software-properties-common`
2. `curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -`
3. `sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"`
4. `apt-cache policy docker-ce`
5. `sudo apt update`
6. `sudo apt install docker-ce`
7. `sudo systemctl status docker`

После установки Docker, требуется установить образ MS SQL Server для x64 систем и выполнить его запуск:

```

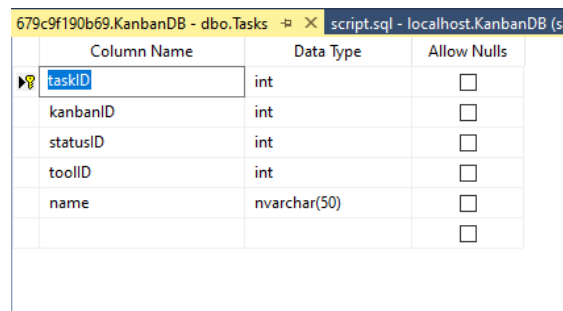
1. docker pull mcr.microsoft.com/mssql/server:2019-CU16-ubuntu-20.04
2. sudo docker run -e "ACCEPT_EULA=Y" -e
   "SA_PASSWORD=yourStrong(!)Password" -e "MSSQL_PID=Express" -p
   1433:1433 --name sqlServer

```

После скачивания и развертывания образа MS SQL Server была выполнена настройка порта SSH.

4.3 Создание базы данных

Создание базы данных производилось методом написания графического дизайна средствами MSSMS, на рисунке 23 показана таблица «Tasks».



Column Name	Data Type	Allow Nulls
taskID	int	<input type="checkbox"/>
kanbanID	int	<input type="checkbox"/>
statusID	int	<input type="checkbox"/>
toolID	int	<input type="checkbox"/>
name	nvarchar(50)	<input type="checkbox"/>

Рисунок 23 – Создание таблицы «Tasks»

С помощью программы SSMS была создана база данных «KanbanDB». В состав этой базы данных были включены 6 взаимосвязанных таблиц. Получившаяся схема базы данных представлена на рисунке 24:

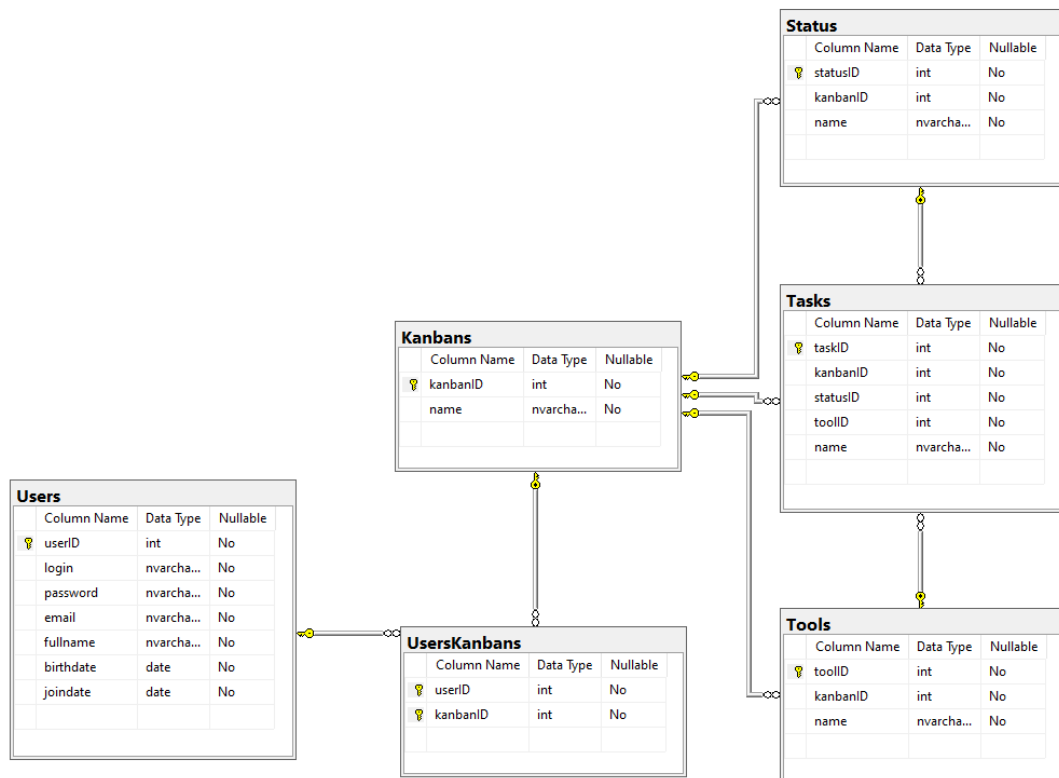


Рисунок 24 – Схема базы данных «KanbanDB»

4.4 Разработка приложения

Для работы приложения была разработана клиентская часть программного обеспечения на языке C# с использованием среды разработки Microsoft Visual Studio 2019. Программа имеет графический интерфейс, разработанный с использованием технологии Windows Forms, и взаимодействует с удалённым MS SQL сервером с использованием технологии Entity Framework.

В разработанной системе существует одна особенность, это способ генерации канбан доски, используя генерацию flowlayoutpanel внутри другого flowlayoutpanel.

На рисунке 25 представлен листинг, генерации во внутреннем flowlayoutpanel, в внутреннем создаются сами карточки задач и название статуса, где они располагаются.

```
public partial class StatusColumn : UserControl
{
    private List<Task> allTasks = new List<Task>(DataBaseContext.database.Tasks);
    private List<Task> needTask = new List<Task>();
    1 reference
    public StatusColumn()
    {
        InitializeComponent();
    }

    1 reference
    public void GenerateStatusColumnData(Status status)
    {
        statusBox.Text = status.name;
        foreach (Task task in allTasks)
        {
            if(task.Status.statusID == status.statusID)
            {
                needTask.Add(task);
            }
        }
        GenerateTaskCardList(needTask);
        allTasks.Clear();
        needTask.Clear();
    }

    1 reference
    private void GenerateTaskCardList(List<Task> taskCards)
    {
        flowLayoutPanel.Controls.Clear();
        foreach (Task task in taskCards)
        {
            TaskCard card = new TaskCard();
            card.GenerateTaskCardData(task);
            flowLayoutPanel.Controls.Add(card);
            card.MouseDoubleClick += new MouseEventHandler(EditTaskCardData);
        }
    }

    1 reference
    private void EditTaskCardData(object sender, EventArgs e)
    {
        TaskCard card = (TaskCard)sender;
        MessageBox.Show(card.Task.name);
    }
}
```

Рисунок 25 – Листинг кода генерации канбана

На рисунке 26 представлен листинг, генерации во внешнем flowlayoutpanel, который отвечает за генерацию каждого внутреннего.

```
public partial class MainForm : Form
{
    List<Kanban> kanbans = new List<Kanban>(DataBaseContext.database.Kanbans.ToList());
    List<Status> allStatuses = new List<Status>(DataBaseContext.database.Status.ToList());
    List<Status> needStatuses = new List<Status>();
    Kanban selectedKanban = new Kanban();
    public User user = null;
    1 reference
    public MainForm(User login)
    {
        InitializeComponent();
        user = login;
    }

    1 reference
    private void MainForm_Load(object sender, EventArgs e)
    {
        kanbanCombo.DataSource = user.Kanbans.Select(k => k.name).ToList();
        kanbanCombo.SelectedIndex = 0;
        labelHello.Text = "Добрый день, " + user.fullname + "!";
        UpdateData();
    }

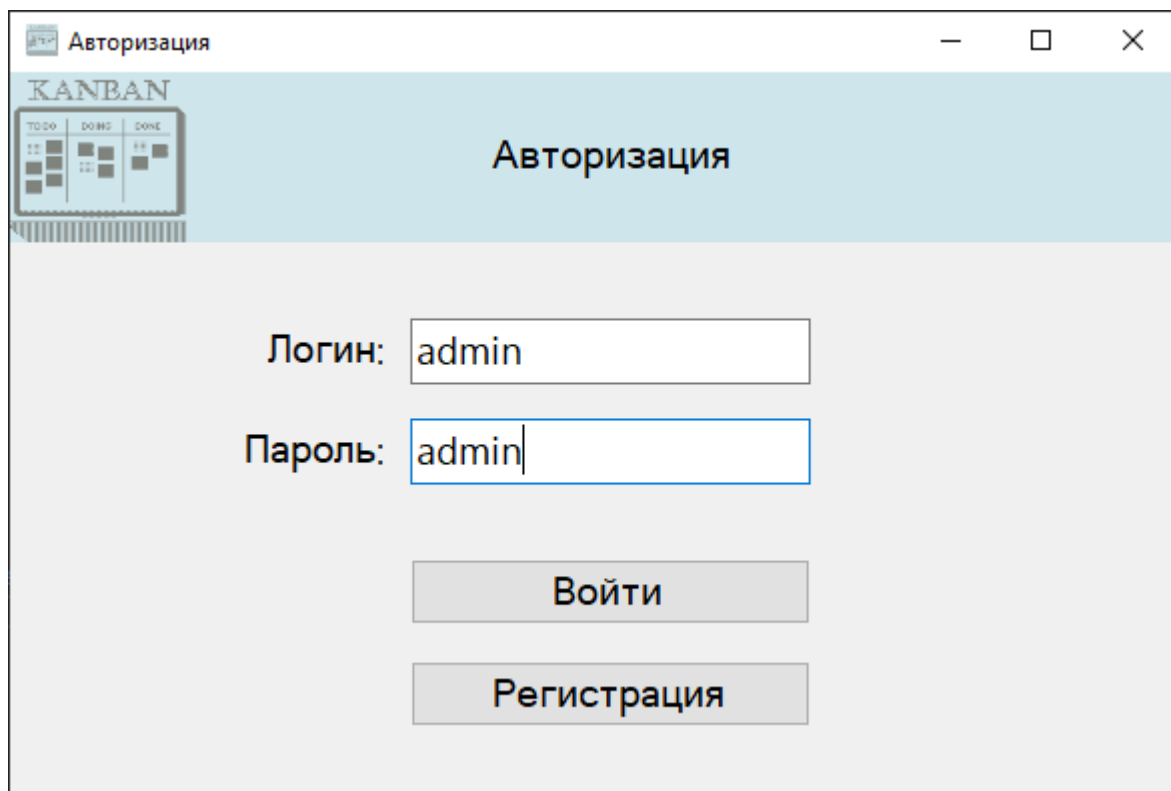
    2 references
    public void UpdateData()
    {
        selectedKanban = kanbans.First(k => k.name.Contains(kanbanCombo.Text));
        foreach (Status status in allStatuses)
        {
            if(status.Kanban.name == selectedKanban.name)
            {
                needStatuses.Add(status);
            }
        }
        GenerateKanban(needStatuses);
        needStatuses.Clear();
    }

    1 reference
    public void GenerateKanban(List<Status> statuses)
    {
        flowKanbanPanel.Controls.Clear();
        foreach(Status status in statuses)
        {
            StatusColumn statusColumn = new StatusColumn();
            statusColumn.GenerateStatusColumnData(status);
            flowKanbanPanel.Controls.Add(statusColumn);
        }
    }

    1 reference
    private void kanbanCombo_SelectedIndexChanged(object sender, EventArgs e)
    {
        flowKanbanPanel.Controls.Clear();
        UpdateData();
    }
}
```

Рисунок 26 – Листинг кода генерации канбана

При первом запуске приложения пользователю открывается окно входа, приведённое на рисунке 27:



Авторизация

Логин: admin

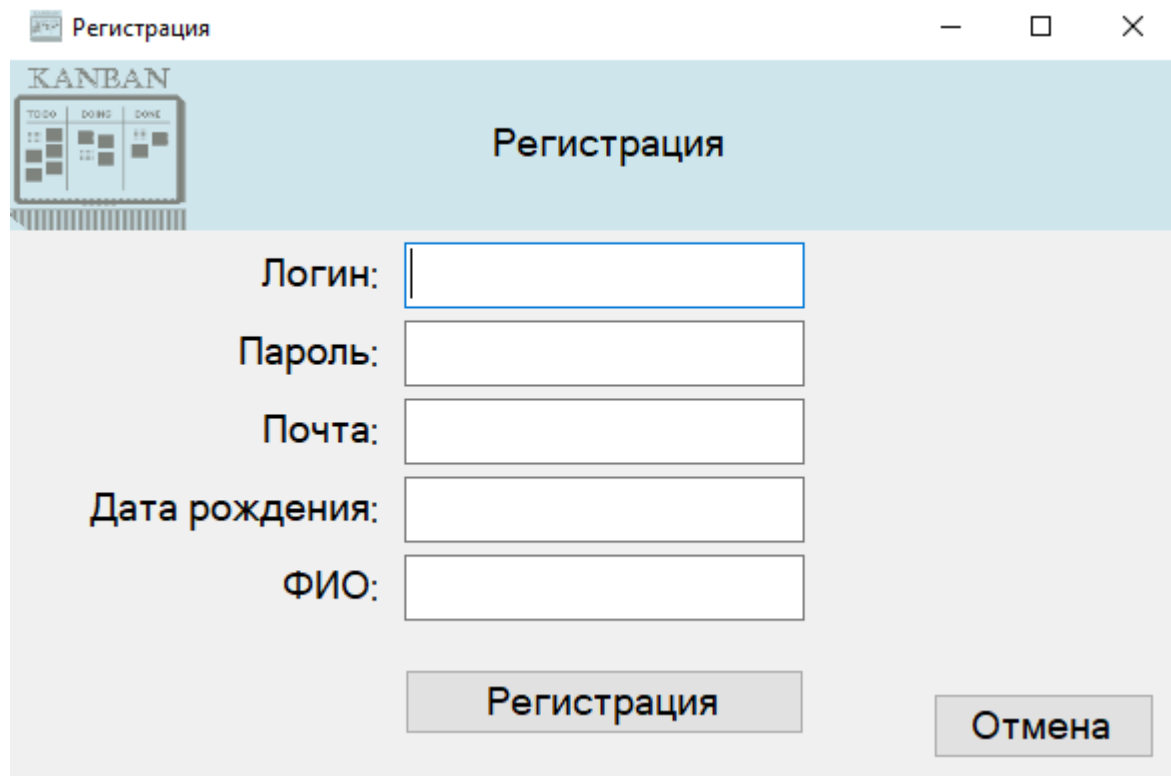
Пароль: admin

Войти

Регистрация

Рисунок 27 – Окно авторизации

Новому пользователю требуется нажать на кнопку «Регистрация» для открытия формы Регистрации, изображенного на рисунке 28:



Регистрация

Логин:

Пароль:

Почта:

Дата рождения:

ФИО:

Регистрация

Отмена

Рисунок 28 – Форма регистрации

После ввода данных выполняется сохранение данных в базу данных, после чего открывается главная форма, приведённая на рисунке 29:

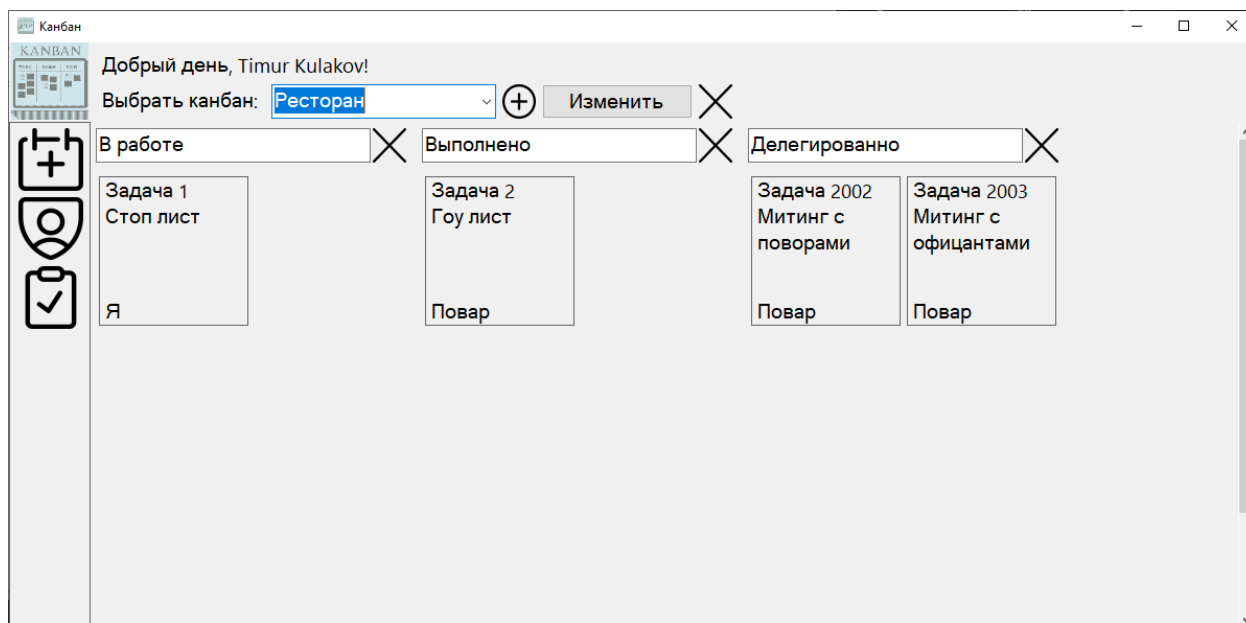


Рисунок 29 – Главное форма приложения

5 Тестирование приложения

5.1 Теоретические сведения

Человеческие ошибки могут приводить к появлению дефектов на всех стадиях разработки программного продукта, причем последствия этого могут быть самыми разными – от незначительных до катастрофических.

Тестирование программного обеспечения — процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определённым образом.

Тестирование позволяет находить и исправлять дефекты, тем самым снижая уровень риска и повышая качество продукта. Проверяются, в том числе, и места пользовательского интерфейса, где пользователь может сделать ошибку или неправильно понять вывод программы, а также устойчивость системы к злонамеренным действиям.

Существует три основных уровня тестирования:

Тестирование компонентов — тестируется минимально возможный для тестирования компонент, например, отдельный класс или функция. Часто тестирование компонентов осуществляется разработчиками программного обеспечения.

Интеграционное тестирование — тестируются интерфейсы между компонентами, подсистемами или системами. При наличии резерва времени на данной стадии тестирование ведётся итерационно, с постепенным подключением последующих подсистем.

Системное тестирование — тестируется интегрированная система на её соответствие требованиям.

По степени автоматизации можно выделить два основных вида тестирования программного обеспечения: ручное и автоматизированное.

Ручное тестирование производится тестировщиком без использования программных средств, для проверки программы или сайта путём моделирования действий пользователя. В роли тестировщиков могут

выступать и обычные пользователи, сообщая разработчикам о найденных ошибках.

Автоматизированное тестирование использует программные средства для выполнения тестов и проверки результатов выполнения, что помогает сократить время тестирования и упростить его процесс. Написание программных средств для автоматизированного тестирования занимает определенный период времени, поэтому автоматизированное тестирование редко применяют для небольших информационных систем.

Регрессионное тестирование – это набор тестов, направленных на обнаружение дефектов в уже протестированных участках приложения. Регрессионное тестирование предназначено для поиска и исправления регрессионных ошибок.

Регрессионные ошибки – те ошибки, которые появляются не при написании программы, а при добавлении нового участка программы или исправлении других ошибок.

При регрессионном тестировании проводится не только тестирование измененных участков, а тестирование всей системы в целом.

5.2 Особенности тестирования в проекте

Особенность тестирования данного проекта состоит в том, что автоматизированное тестирование применять к нему нерационально. Необходимо провести ручное функциональное тестирование. После исправления всех ошибок должно быть проведено регрессионное тестирование.

5.3 Отчёт о тестировании

Для тестирования приложения был сформирован набор позитивных и негативных тест-кейсов, которые были проведены в системе. Тесты обеспечивают проверку всей функциональности системы. Результаты тестирования по составленным тестам занесены в таблицы.

5.3.1 Позитивные тест-кейсы

Позитивные тест-кейсы используют только корректные (ожидаемые программой) данные и проверяют, что приложение правильно выполняло вызываемую функцию. Список позитивных тест-кейсов для приложения «KanbanApp» приведён на таблицах 6-8:

Таблица 6 – Первый набор тест-кейсов окна авторизации

Название поля ввода	Тест 1	Тест 2
Логин	admin	user
Пароль	admin	user
Ожидаемый результат	Осуществление входа в приложение от лица пользователя Admin.	Осуществление входа в приложение от лица пользователя user.
Фактический результат	Осуществление входа в приложение от лица пользователя admin.	Осуществление входа в приложение от лица пользователя user.
Результат тест-кейса	Пройден	Пройден

Таблица 7 – Тестирование окна регистрации

Название поля ввода	Тест 3	Тест 4
Логин	povar	Aleksey332
Пароль	123456povar	PasswordPassword
ФИО	Анатолий Хопкин Андреевич	Алексей Сержоу Сергеевич
Дата рождения	19.06.1986	25.01.2000
E-mail	povar@gmail.com	ivanovIv@yandex.ru
Ожидаемый результат	Сохранение данных; Переход на окно авторизации	Сохранение данных; Переход на окно авторизации
Фактический результат	Сохранение данных; Переход к авторизации	Сохранение данных; Переход к авторизации
Результат тест-кейса	Пройден	Пройден

Таблица 8 – Первый набор тест-кейсов для окна добавление задачи

Название поля ввода	Тест 5	Тест 6
Канбан	Ресторан	Отчетность
Статус	Выполнено	Делегировано
Соисполнитель	Повар	Бухгалтер
Название	Митинг с официантами	Ежедневный отчет
Ожидаемый результат	Сохранение данных;	Сохранение данных;
Фактический результат	Сохранение данных;	Сохранение данных;
Результат тест-кейса	Пройден	Пройден

5.3.2 Негативные тест-кейсы

Негативные тест-кейсы – ввод некорректных данных, которых программа не ожидает. Например, если используется неверный регистр. Список негативных тест-кейсов для приложения «KanbanApp» приведён на таблице 9:

Таблица 9 – Второй набор тест-кейсов окна авторизации

Название поля ввода	Тест 13	Тест 14
Логин	admin	AdMiN
Пароль		admin
Ожидаемый результат	Вывод сообщения «Заполните поля: Логин, Пароль»	Вывод сообщения «Несуществующий логин»
Фактический результат	Вывод сообщения «Заполните поля: Логин, Пароль»	Вывод сообщения «Несуществующий логин»»
Результат тест-кейса	Пройден	Пройден

6 Техника безопасности и охрана труда

6.1 Вредоносные факторы на стадии разработки приложения

Правила выполнения работ с ПК обязаны обеспечивать минимизацию основных рисков, перечисленных в ГОСТ. В первую очередь к ним относятся:

- повышенная температура отдельных элементов персонального компьютера;
- монотонность трудового процесса;
- высокий уровень зрительных нагрузок;
- вероятность поражения статическим электричеством, генерируемым техникой;
- недостаточная освещенность в зоне выполнения операций у компьютера;
- высокий уровень напряженности электрического и магнитного полей в зоне установки компьютера;
- другие факторы.

6.2 Требования охраны труда перед началом работы

Перед началом работы сотрудник должен:

- Проветрить помещение, устранить повышенную подвижность воздуха (сквозняки) и т.п.
- Отрегулировать освещенность на рабочем месте и убедиться в ее достаточности.
- Проверить правильность расположения элементов компьютера в целях исключения неудобных поз и длительных напряжений тела.
- Убрать с рабочего места посторонние предметы, освободить подходы к рабочему месту.
- Убедиться путем внешнего осмотра:
 - в исправности кабельных соединений, проводов, вилок, розеток;

- в том, что кабели электропитания ПК и другого оборудования (включая переноски и удлинители) находятся с тыльной стороны рабочего места;
- в максимальной удаленности от рабочего места источника бесперебойного питания для исключения вредного влияния его повышенных магнитных полей.

6.3 Требования охраны труда во время работы

- Не допускать натягивания, скручивания, перегиба и пережима шнуров электропитания ПК, не допускать нахождения на них каких - либо предметов и соприкосновения их с нагретыми поверхностями.
 - Не допускать попадания влаги на поверхность персонального компьютера.
 - Не прикасаться к задней панели системного блока при включенном питании.
 - Не оставлять включенный ПК без наблюдения.
 - Не производить самостоятельно какие-либо виды ремонта ПК.
 - Не закрывать вентиляционные отверстия системного блока ПК.
 - При работе на ПК соблюдать расстояние от глаз до экрана в пределах
 - 60-70 см, но не ближе 50 см. с учетом размеров алфавитно-цифровых знаков и символов.
 - Продолжительность непрерывной работы на ПК без регламентированного перерыва не должна превышать двух часов.
 - Содержать свободными проходы к рабочему месту.
 - Не загромождать рабочее место.
- ### **6.4 Требования охраны труда в аварийных ситуациях**
- При возникновении аварийной ситуации на рабочем месте необходимо:

- немедленно прекратить работу

- сообщить о возникновении аварийной ситуации непосредственному руководителю или лицу, его замещающему;
- под руководством непосредственного руководителя принять участие в ликвидации создавшейся аварийной ситуации, если это не представляет угрозы для здоровья и жизни работников;
- при необходимости покинуть опасную зону.
- При возникновении нарушений в работе персонального компьютера, а также при возникновении нарушений в работе электросети (запах гари, дым, посторонний шум при работе ПК, ощущение действия электрического тока при прикосновении к элементам ПК и т.п.) необходимо:
 - прекратить работу;
 - отключить ПК от электросети;
 - сообщить о случившемся непосредственному руководителю или лицу его замещающему.
 - Не приступать к работе до полного устранения повреждений или неисправностей.
- При возникновении возгорания необходимо:
 - прекратить работу;
 - отключить ПК от электросети;
 - сообщить о случившемся непосредственному руководителю или лицу его замещающему;
 - приступить к тушению возгорания углекислотным или порошковым огнетушителем;
 - при невозможности ликвидировать возгорание, покинуть опасную зону через эвакуационные выходы в соответствии со схемой эвакуации.
- При резком ухудшении состояния здоровья или травмировании необходимо прекратить работу, сообщить о случившемся непосредственному

руководителю или лицу его замещающему. При необходимости воспользоваться аптечкой.

- При несчастном случае во время работы:
 - оказать пострадавшему первую помощь в соответствии с Инструкцией по оказанию первой (доврачебной) помощи при несчастном случае;
 - при необходимости доставить пострадавшего в ближайшее медицинское учреждение или вызвать скорую медицинскую помощь;
 - немедленно сообщить о случившемся своему непосредственному руководителю или лицу его замещающему;
 - принять меры для сохранения обстановки несчастного случая, если это не сопряжено с опасностью для жизни и здоровья.

Заключение

В ходе выполнения дипломного проекта по разработке приложения для планирования распорядка дня для руководителя ресторана были выполнены ряд задач.

Проведен анализ предметной области и моделирование бизнес-процессов руководителя ресторана с использованием бережливого производства с построением организационных диаграмм и функциональных моделей предметной области в нотации IDEF.

В результате проведения анализа предметной области были выработаны основные направления разработки и функционал разрабатываемого приложения. С помощью нотации UML была представлена концептуальная модель информационной системы. Проведено объектно-ориентированное моделирование, включающие построение основных диаграмм, а именно: диаграмма прецедентов, диаграммы деятельности и последовательности, диаграмма кооперации, а также диаграмма «сущность-связь», представляющая информационную (логическую) модель ИС.

Был проведен анализ интерфейса существующих приложений для планировки дня. Разработаны и представлены макеты приложения.

Подсистема была разработана в виде приложения для терминала на языке C# с помощью программного средства Visual Studio 2019 с подключенной к ней базой данных. База данных была разработана с помощью программного средства SQL Server Management Studio. Также представлен графический интерфейс по построенным макетам, разработанных с использованием технологий Windows Forms. Подсистема полностью функционирует и позволяет работать в ней работникам склада театральных декораций.

Было проведено ручное тестирование подсистемы, для выявления ошибок, допущенных на стадии разработки. В ходе тестирования были обнаружены ошибки приложения, которые были устранены. Так же

тестирование позволило проверить полную работоспособность приложения. Результаты тестирования представлены в виде таблиц баг-репортов.

В результате дипломного проектирования разработано приложения для руководителя ресторана, программа получилась простой для понимания, с удобным и понятным интерфейсом.

Все материалы дипломного проекта сформированы и оформлены согласно требованиям методических рекомендаций по оформлению текстовых документов ГБПОУ «Первый Московский Образовательный Комплекс».

Список используемых источников

1. ГОСТ 19.701-90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения.
2. ГОСТ 2.105-95 Единая система конструкторской документации (ЕСКД). Общие требования к текстовым документам.
3. ГОСТ 28195-89 Оценка качества программных средств. Общие положения.
4. ГОСТ 28397-89 Языки программирования. Термины и определения.
5. Аллен Эрик, Типичные ошибки проектирования / Эрик Аллен – Питер – Санкт-Петербург, 2016. – 224 с.
6. Виссер Дж. Разработка обслуживаемых программ на языке C# / пер. с англ. Р. Н. Рагимова. — М.: ДМК Пресс, 2017. — 192 с.: ил.
7. Дейв Томас, Энди Хант. Программист-прагматик: Второе юбилейное издание: Пер. с англ. – СПб. : ООО «Диалектика», 2020 – 368 с. : ил. — Парал. тит. Англ.
8. Дебаррос Энтони, Practical SQL: A Beginner's Guide to Storytelling with Data / Энтони Дебаррос. – No Starch Press – Сан-Франциско, 2018 – 527 с.
9. Джапиксе Филип, Pro C# 9 with .NET 5: Foundational Principles and Practices in Programming / Филип Джапиксе, Эндрю Троелсен. – Нью-Йорк – Apress, 2021 – 1411 с.
10. Левицкий Н. Д. Удаленный сервер своими руками. От азов создания до практической работы. / Н. Д. Левицкий – Наука и Техника – Санкт-Петербург, 2021 – 400 с.
11. Мартин Р. К. Чистая архитектура. Искусство разработки программного обеспечения / Р. К. Мартин. – Питер – Санкт-Петербург, 2018. – 352 с.
12. Мартин Р. К. Чистый код. Создание, анализ и рефакторинг / Р. К. Мартин. – Питер – Санкт-Петербург, 2016. – 464 с.

13. Кригель, Алекс, Трухнов, Борис. K82 SQL. Библия пользователя, 2-е издание.: Пер. с англ. — М.: ООО “И.Д. Вильямс”, 2017. — 752 с.: ил. — Парал. тит. Англ.

14. Майерс Г., Баджетт Т. Искусство тестирования программ. Пер. с англ. — М.: Издательский дом «Вильямс», 2017. — 272 с.

15. Молинаро Э. SQL. Сборник рецептов. — Пер. с англ. — СПб: Символ-Плюс, 2009. — 672 с., ил.

16. Симонович С. В. Информатика. Базовый курс: Учебник для вузов. 3-е изд. Стандарт третьего поколения. — СПб.: Питер, 2018. — 640 с.: ил.

17. Скит, Джон. C# для профессионалов: тонкости программирования, 3-е изд.: Пер. с англ. — М.: ООО “И.Д. Вильямс”, 2017. — 608 с.: ил. — Парал. тит. англ.

18. Фуфаев Д.Э. Разработка и эксплуатация автоматизированных информационных систем: учебник для студ. сред. проф. образования. М.: Издательский центр «Академия», 2019.- 304 с.

19. Пайпер Бен, Learn Cisco Network Administration in a Mounth of Lunches / Бен Пайпер. — ДМК Пресс — Москва, 2018 — 316 с.

20. Закон РФ от 07.02.1992 N 2300-1 (ред. от 11.06.2021) "О защите прав потребителей"

21. 1cloud [Электронный ресурс] // Режим доступа: <https://1cloud.ru>.

22. Cyber01 [Электронный ресурс] // Режим доступа: <https://cyber01.ru>.

23. CyberForum.ru [Электронный ресурс] // Режим доступа: <https://www.cyberforum.ru>.

24. Database.Guide [Электронный ресурс] // Режим доступа: <https://database.guide>.

25. Ubuntu Desktop Guide [Электронный ресурс] // Режим доступа: <https://help.ubuntu.com/stable/ubuntu-help/>.

26. ИНТУИТ. Национальный открытый университет. Информационные системы [Электронный ресурс] <https://intuit.ru/studies/courses/3627/869/lecture/31731>

27. ИНТУИТ. Национальный открытый университет. Проектирование информационных систем: Информация. [Электронный ресурс]
<https://intuit.ru/studies/courses/2195/55/info>

28. ИНТУИТ. Национальный открытый университет. Проектирование информационных систем: Информация. [Электронный ресурс]
<https://intuit.ru/studies/courses/2195/55/info>

29.

30. Information Security Squad [Электронный ресурс] // Режим доступа:
<https://itsecforu.ru>.

31. Losst [Электронный ресурс] // Режим доступа: <https://losst.ru>.

32. VITUX [Электронный ресурс] // Режим доступа: <https://vitux.com/>

33. Комьюнити. [Электронный ресурс] // Режим доступа:
<https://timeweb.com/ru/community>.

34. Хабр [Электронный ресурс] // Режим доступа: <https://habr.com/ru/all>.

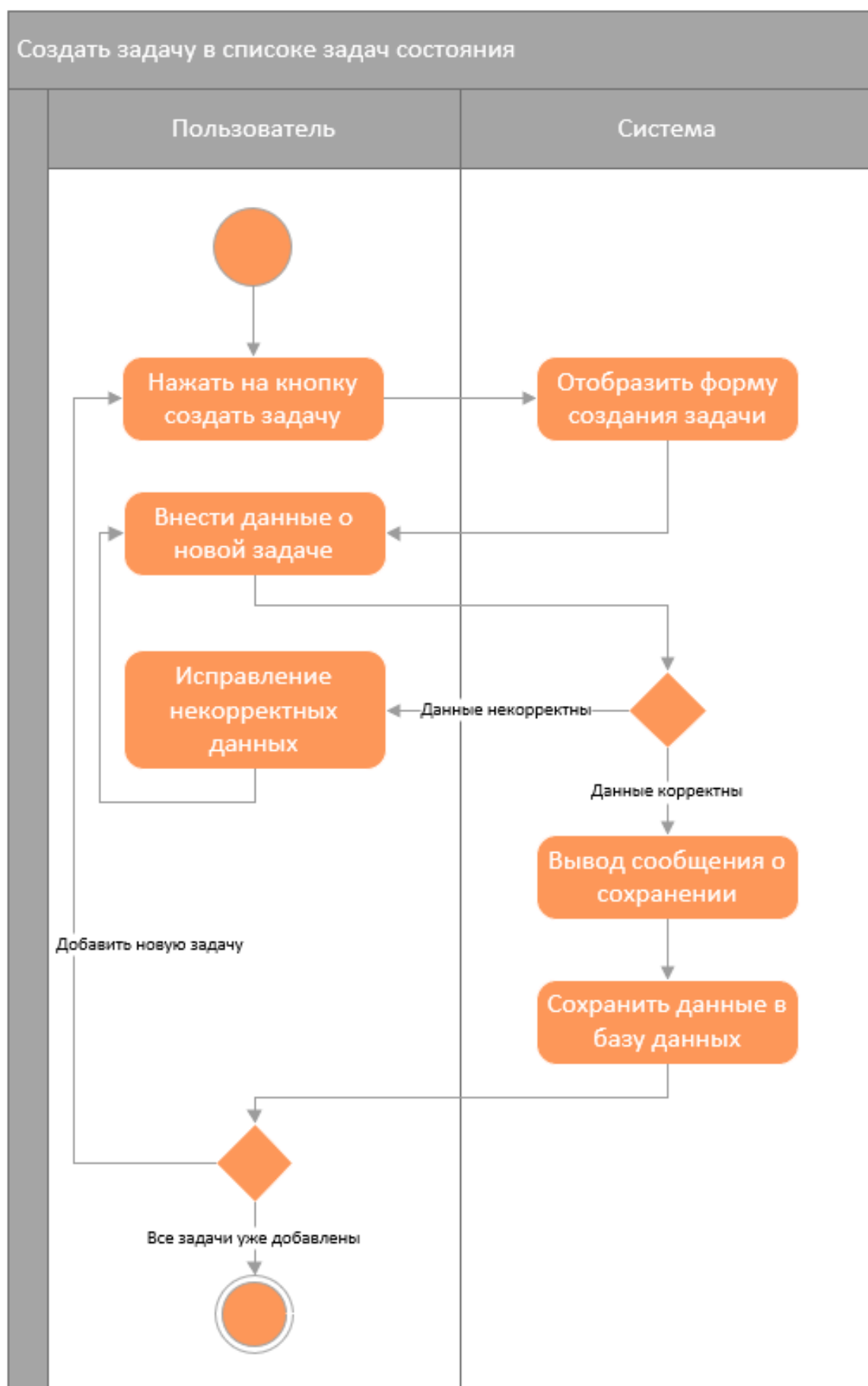


Рисунок 1 – Диаграмма деятельности