

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

вычислительной техники

Кафедра

МОиПЭВМ

Направление 09.03.02 «Информационные системы и технологии»
подготовки

БАКАЛАВРСКАЯ РАБОТА

на тему

**«Информационная система контроля и учета рабочего
времени сотрудников компании»**

Студент

(подпись, дата) Богатов Кирилл Алексеевич
(ФИО полностью)

Руководитель

(подпись, дата) Кольчугина Е.А.
(фамилия, инициалы)

Нормоконтролёр

(подпись, дата) Такташкин Д.В.
(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой

(подпись) Козлов А.Ю.
(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК

(подпись) Попова Н.А.
(фамилия, инициалы)

Пенза, 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Кафедра

Вычислительной техники

МО и ПЭВМ

«Утверждаю»

Заведующий кафедрой

А.Ю. Козлов

«25» декабря 2023 г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА

1. Студент Богатов Кирилл Алексеевич гр. 20ВИ2 факультета вт
<Фамилия, имя, отчество полностью>

2. Тема работы «Информационная система контроля и учета рабочего времени сотрудников компании».

Тема утверждена приказом ПГУ № _____ от " _____ " _____ 20 _____

2. Руководитель работы Кольчугина Е.А.

3. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать информационную систему контроля и учета рабочего времени сотрудников компании. Среда и средства разработки: C#, ASP.NET, .NET Framework 4.8, MySqlGDB, JSONExtended, MySql + phpMyAdmin. Среда функционирования: Windows 7 и выше. Основные функции: осуществление контроля рабочего времени сотрудников, создание проектов и распределение их между сотрудниками, установка задач и контроль выполнения задач подразделениями и каждым,

сотрудником отдельно, учет рабочего времени сотрудника в системе авторизация сотрудника в системе. Особенности (особые функции, особые требования): Бесплатное ПО, Отсутствуют легкие аналоги не полных CRM систем с реализованной многозадачностью, отсутствуют CRM системы с поддержкой выполнения задач подразделениями, наличие уведомлений

4. Перечень подлежащих разработке вопросов.

1. Анализ предметной области и постановка задачи.

2. Анализ требований на разработку.

3. Разработка программных средств.

4. Тестирование программных средств.

5. Оформление ВКР.

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области и постановка задачи	15%	08.04.2024-13.04.2024	
2.	Анализ требований на разработку	20%	15.04.2024-25.04.2024	
3.	Разработка программных средств	30%	26.04.2024-12.05.2024	
4.	Тестирование программных средств	20%	13.05.2024-22.05.2024	
5.	Оформление ВКР	15%	23.05.2024-03.06.2024	

Дата выдачи задания "21" декабря 2023г.

Руководитель бакалаврской работы _____

(подпись, дата)

Задание к исполнению принял студент _____

(подпись, дата)

Работу к защите допустить

Декан факультета _____

(подпись, дата)

Реферат

Пояснительная записка содержит 75 листов, 18 рисунков, 4 таблицы, 15 использованных источников, 7 листингов и 3 приложения.

АИС, ASP.NET, VISUAL STUDIO 2022, C#, УЧЕТ ВРЕМЕНИ, ИНФОРМАЦИОННАЯ СИСТЕМА.

Предмет работы: разработка информационной системы автоматизации контроля рабочего времени сотрудников.

Объект разработки: клиент серверное приложение, обеспечивающее деятельность контроля времени сотрудников.

Цель работы: разработка информационной системы, которая автоматизирует деятельность контроля рабочего времени сотрудников.

Средства разработки: C#, ASP.NET, .NET Framework 4.8, MySqlGDB, JSONExtended, MySql.

Результаты работы: спроектировано, разработано, отлажено и протестировано клиент-серверное приложение учета и контроля времени сотрудников.

Область применения: любые компании, деятельность сотрудников которых требуется отслеживать.

					ПГУ 09.03.02 – 08ВР202.12 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	«Информационная система контроля и учета рабочего времени сотрудников компании» Пояснительная записка	Лит.	Лист	Листов
Разраб.		Богатов К.А.					4	75
Провер.		Кольчугина Е.А.						
Реценз.								
Н. Контр.		Такташкин Д.В.						
Утверд.		Козлов А.Ю.				группа 20ВИ2		

Содержание

Введение.....	11
1 Аналитическая часть.....	12
1.1 Анализ предметной области	12
1.1.1 Требования к составу и параметрам технических средств.....	13
1.1.2 Описание функциональности контроля рабочего времени	13
1.2 Анализ информационных систем аналогичного назначения	14
1.2.1 «Битрикс24».....	14
1.2.2 «Shtab».....	15
1.2.3 «АmoCRM».....	17
1.2.4 Обоснование для разработки клиент серверного приложения системы контроля рабочего времени сотрудников	18
1.3 Анализ требований к программной реализации.....	20
1.3.1 Список функциональных требований.....	20
1.3.2 Список нефункциональных требований.....	22
1.3.3 Требования к надежности	22
1.3.4 Условия эксплуатации.....	22
2 Проектирование программного обеспечения	23
2.1 Проектирование пользовательского интерфейса	23
2.2 Анализ бизнес процессов организации в нотации IDEF0	25
2.3 Проектирование модели базы данных.....	26
2.3.1 Проектирование физической и инфологической моделей базы данных с помощью Erwin	27
2.4 Проектирование поведения ИС.....	29
2.4.1 Диаграмма состояний	30
2.4.2 Диаграмма деятельности.....	30
2.4.3 Диаграмма последовательности	31
2.4.4 Диаграмма коммуникаций	31

2.4.5	Диаграмма компонентов	32
2.4.6	Диаграмма развертывания.....	32
3	Разработка приложения.....	34
3.1	Программные средства.....	34
4	Отладка и тестирование	37
4.1	Тестирование программного обеспечения.....	37
4.1.1	Метрики программного средства.....	41
4.1.2	Расчет метрик кода с помощью встроенного средства Visual Studio	41
4.1.3	Определение метрик компонента серверной части приложения.....	42
	Заключение	45
	Список использованных источников	46
	Приложение А (рекомендуемое) Диаграммы UML	48
	Приложение Б (рекомендуемое) Листинг программы	54
	Приложение В (рекомендуемое) Результаты тестирования.....	74

Введение

Эффективное управление рабочим временем сотрудников играет ключевую роль в успешном функционировании любой компании. Создание информационной системы контроля и учета рабочего времени – это не просто обеспечение правильного распределения трудовых ресурсов, но и стратегический шаг в повышении производительности и оптимизации бизнес-процессов.

В наше время, когда интернет стал неотъемлемой частью нашей повседневной жизни, эффективная информационная система контроля и учета рабочего времени становится особенно важной. Подобно тому, как клиенты и партнеры ищут информацию о компании и её продуктах в сети, так и эффективная система учета рабочего времени помогает организации лучше понимать, как эффективно используются трудовые ресурсы.

Программное обеспечение для контроля и учета рабочего времени представляет собой целый комплекс функций и возможностей. В его основе – клиент-серверная архитектура, которая позволяет собирать и анализировать данные о рабочем времени сотрудников в режиме реального времени.

Система включает в себя функционал, позволяющий:

- регистрировать время прихода и ухода сотрудников;
- отслеживать время проведенное на рабочем месте и вне его;
- автоматически вычислять отработанные часы и сверять их с графиком работы;
- предоставлять администратору возможность управлять данными и генерировать отчеты о рабочем времени.

Добавление, удаление и изменение информации о сотрудниках и их рабочем времени производится через административный интерфейс системы, доступный только уполномоченным пользователям.

1 Аналитическая часть

1.1 Анализ предметной области

Администратор информационной системы контроля и учета рабочего времени – это ключевая фигура, занимающаяся управлением данными о рабочем времени сотрудников компании. Он отвечает за добавление, удаление и изменение информации о сотрудниках, а также за обеспечение корректной работы системы.

Проектируемая система предназначена для хранения информации о рабочем времени сотрудников, а также для формирования отчетов и анализа производительности труда.

Система создается для обслуживания следующих групп пользователей:

- руководство компании – получает отчеты о рабочем времени и производительности сотрудников;
- HR-отдел – осуществляет управление данными о сотрудниках и их графиках работы;
- сотрудники – могут просматривать свои графики работы и вносить изменения при необходимости.

Информационная система должна обеспечивать:

- ведение базы данных о рабочем времени сотрудников (запись, чтение, модификация, удаление);
- хранение информации о графиках работы, отработанных часах и прочих данных о рабочем времени;
- возможность генерации отчетов и анализа производительности труда сотрудников.

Цель разработки заключается в создании надежной и удобной информационной системы, которая позволит компании эффективно управлять рабочим временем своих сотрудников, повысить производительность труда и оптимизировать бизнес-процессы.

1.1.1 Требования к составу и параметрам технических средств

Для работы с Web-приложением рекомендуются следующие программные и технические средства:

- операционная система семейства Windows/Linux;
- современный закупочный браузер (рекомендуются Google Chrome, Firefox);
- одноядерный процессор Intel или совместимый с частотой ядра не менее 500МГц;
- объем свободной оперативной памяти 512 Мб;
- объем необходимой памяти на жестком диске 500 Мб;
- стандартный VGA-монитор или совместимый, с поддерживаемым разрешением от 1024x768;
- 3D-ускоритель с объемом оперативной памяти больше 32 Мб;
- стандартная клавиатура;
- манипулятор «мышь».

1.1.2 Описание функциональности контроля рабочего времени

В настоящее время существует много информационных систем контроля рабочего времени сотрудников.

Внедрение автоматизированной информационной системы позволит:

- снизить случайные ошибки связанных с ручным отслеживанием задач;
- автоматизировать обработку информации в системе;
- получить доступ к списку задач сотруднику из любой точки, где есть компьютер, имеющий выход в Интернет;
- автоматически контролировать выполнение дневных и общих задач сотрудниками.

1.2 Анализ информационных систем аналогичного назначения

1.2.1 «Битрикс24»

Bitrix24 CRM – это мощный инструмент управления отношениями с клиентами, специально адаптированный для компаний в любой сфере, где требуется следить за временем сотрудников. Он предлагает широкий набор функций, позволяющих эффективно контролировать рабочее время сотрудников, оптимизировать бизнес-процессы и обеспечивать высокий уровень обслуживания клиентов. Bitrix24 CRM помогает компаниям, подобным вашей, удерживать лидирующие позиции на рынке и успешно развиваться.

Функции управления графиками работы:

- запись и редактирование информации о рабочем времени сотрудников (время начала и окончания работы, перерывы);
- категоризация сотрудников по должностям, отделам, проектам;
- установка целей и планирование рабочего времени.

Мониторинг производительности:

- отслеживание отработанных часов и выполненных задач;
- анализ эффективности работы сотрудников и выявление возможностей для улучшения производительности;
- генерация отчетов о рабочем времени и производительности сотрудников.

Функции управления проектами:

- создание проектов и назначение задач сотрудникам;
- мониторинг выполнения задач и контроль сроков выполнения;
- совместная работа над проектами и обмен информацией внутри команды.

Преимущества использования Bitrix24 CRM для контроля рабочего времени сотрудников:

- эффективное управление рабочим временем: система помогает оптимизировать расписание сотрудников и повышает производительность труда;
- мониторинг производительности: постоянный анализ работы сотрудников позволяет выявлять проблемные моменты и улучшать процессы;
- улучшение коммуникации: Bitrix24 CRM предоставляет инструменты для эффективного взаимодействия и обмена информацией внутри команды;
- гибкость и масштабируемость: система легко настраивается под нужды вашего бизнеса и может быть масштабирована в соответствии с его ростом.

Bitrix24 CRM – это незаменимый инструмент для компаний в любой сфере, которые стремятся к эффективному управлению бизнесом и повышению конкурентоспособности. Благодаря своему функционалу и удобному интерфейсу, Bitrix24 CRM помогает оптимизировать бизнес-процессы, повышать производительность и увеличивать прибыль.

1.2.2 «Shtab»

«Shtab» CRM – это инновационная система управления рабочим временем сотрудников, разработанная специально для компаний в сфере строительства и ремонта. Она предоставляет широкий спектр функций, которые позволяют эффективно контролировать рабочее время, оптимизировать процессы и повышать производительность. «Shtab» помогает компаниям в этой области укреплять свои позиции на рынке и обеспечивать высокое качество услуг для клиентов. Данная система обладает обширным функционалом.

Функции управления графиками работы:

- внесение и корректировка данных о начале и окончании рабочего дня сотрудников;
- гибкое планирование графиков работы сотрудников с учетом специфики проектов и заказов;
- установка целей и контроль выполнения задач.

Мониторинг производительности:

- отслеживание отработанных часов и выполненных задач;
- анализ эффективности работы сотрудников и выявление узких мест в процессах;
- генерация отчетов о производительности и использовании ресурсов.

Функции управления проектами:

- создание и назначение задач по проектам;
- мониторинг выполнения задач и контроль сроков;
- обмен информацией и документами внутри команды.

Преимущества использования системы «Shtab» для управления рабочим временем:

- эффективное распределение ресурсов: Shtab помогает оптимизировать загрузку сотрудников и улучшить использование рабочего времени;
- мониторинг производительности: постоянный контроль позволяет быстро реагировать на изменения и улучшать процессы;
- улучшение коммуникации: система предоставляет инструменты для удобного обмена информацией и совместной работы;
- гибкость и масштабируемость: Shtab легко настраивается под конкретные потребности компании и может масштабироваться в соответствии с ее ростом.

«Shtab» CRM – это незаменимый инструмент для компаний в сфере строительства и ремонта, которые стремятся к повышению эффективности бизнеса и улучшению качества предоставляемых услуг. Благодаря своим возможностям и удобному интерфейсу, «Shtab» помогает оптимизировать процессы, повышать производительность и увеличивать прибыль.

1.2.3 «АmoCRM»

«АmoCRM» – это интегрированная система управления рабочим временем сотрудников, разработанная специально для компаний в сфере маркетинга и продаж. Она предлагает широкий спектр функций, которые позволяют эффективно контролировать рабочее время, оптимизировать процессы взаимодействия с клиентами и повышать эффективность командной работы. «АmoCRM» помогает компаниям в этой области укреплять свои позиции на рынке и обеспечивать высокий уровень обслуживания клиентов.

Функции управления графиками работы:

- запись и редактирование данных о рабочем времени сотрудников (время начала и окончания рабочего дня, перерывы);
- гибкое планирование графиков работы сотрудников с учетом их текущих задач и проектов;
- установка целей и контроль выполнения задач в рамках определенного временного периода.

Мониторинг производительности:

- отслеживание отработанных часов и выполненных задач;
- анализ эффективности работы сотрудников и выявление потенциальных узких мест в процессах;
- генерация отчетов о производительности и использовании рабочего времени.

Функции управления задачами и проектами:

- создание и назначение задач для сотрудников;

- мониторинг выполнения задач и контроль сроков;
- совместная работа над проектами и обмен информацией внутри команды.

Преимущества использования системы «АmoCRM» для управления рабочим временем:

- эффективное управление временем: АmoCRM позволяет оптимизировать расписание работы сотрудников и повысить производительность труда;
- мониторинг производительности: постоянный анализ работы позволяет быстро реагировать на изменения и оптимизировать рабочие процессы;
- улучшение коммуникации: система предоставляет инструменты для эффективного взаимодействия и совместной работы внутри команды;
- гибкость и масштабируемость: АmoCRM легко настраивается под индивидуальные потребности компании и может масштабироваться в соответствии с ее ростом.

«АmoCRM» – это незаменимый инструмент для компаний в сфере маркетинга и продаж, которые стремятся к повышению эффективности бизнеса и улучшению взаимодействия с клиентами. Благодаря своим функциональным возможностям и удобному интерфейсу, «АmoCRM» помогает оптимизировать процессы, повышать производительность и увеличивать прибыль.

1.2.4 Обоснование для разработки клиент серверного приложения системы контроля рабочего времени сотрудников

В рассмотренных аналогичных системах после анализа выявлено следующее:

- внешний вид и устройство сайтов не позволяют пользователю сходу освоиться и начать выполнять задачи, указанные в списке задач;

- для созданий новых задач зачастую нужна регистрация пользователя и создание личного кабинета;
- устаревшая система контроля над временем сотрудников.

Лучшим решением выше выявленных сложностей является разработка автоматизированной информационной системы с использованием технологии ASP.NET [1], потому что ASP.NET – это один из самых популярных фреймворков для разработки сайтов. Инфраструктура ASP.NET MVC [5] представляет собой последнюю версию веб-платформы ASP.NET от Microsoft. У нее есть множество преимуществ, которых нет в классической платформе для веб-разработки ASP.NET Web Forms.

Встроенные методы представлений генерируют ясный и соответствующий стандартам код разметки в формате HTML. Таким образом ASP.NET MVC [5] предлагает мощную систему маршрутизации URL, позволяющую создавать удобные и приятные в смысловом плане URL-адреса. Компоненты ASP.NET MVC расширяемы и тестируемы, она обеспечивает расширенную поддержку разработки через тестирование. Огромным преимуществом является также мощная полнофункциональная среда разработки – Microsoft Visual Studio.

Для платформы .NET на сегодняшний день существуют сотни тысяч бесплатных библиотек и компонентов, доступных для скачивания. Также присутствует удобная система документации и база данных чуть ли не по всем вопросам, которые могут возникнуть в процессе разработки, что так же является огромным плюсом в сторону данного инструмента разработки.

Ожидаемый результат разработки системы:

- современность – клиент-серверное приложение, разработанное с помощью современных инструментов разработки;
- понятный интерфейс – имеется ввиду то, что пользователь сможет без особых усилий понять, как функционируют элементы системы и для чего они нужны;

- комфорт – пользователь сможет просматривать каталог задач и оформлять заявку без необходимости совершения лишних действий, в том числе и регистрации;
- непротиворечивость и достоверность данных;
- низкая стоимость – так же является неоспоримым преимуществом.

1.3 Анализ требований к программной реализации

Спроектированная и полностью разработанная АИС [2] является целью выпускной квалификационной работы.

После разработки и последующего внедрения АИС предполагаются следующие достоинства:

- улучшение эффективности выполнения функций;
- неизменный контроль над процессом формирования и управление рабочим расписанием;
- увеличение количества категорий задач;
- добавление информационных и справочных разделов.

1.3.1 Список функциональных требований

Автоматизированная информационная система учета рабочего времени сотрудников должна выполнять необходимый функционал:

- ведение базы данных (запись, чтение, модификация, удаление);
- хранение информации о расписании и задачах;
- временное хранение информации о содержимом задач на день;
- контроль администратора над выполнением задач.

Для описания поведения и определения функциональных требований разрабатываемой системы существует UML диаграмма прецедентов, состоящая из актеров (действующих лиц), которые взаимодействуют с системой по средствам вариантов использования.

Диаграмма вариантов использования представлена на рисунке 1.

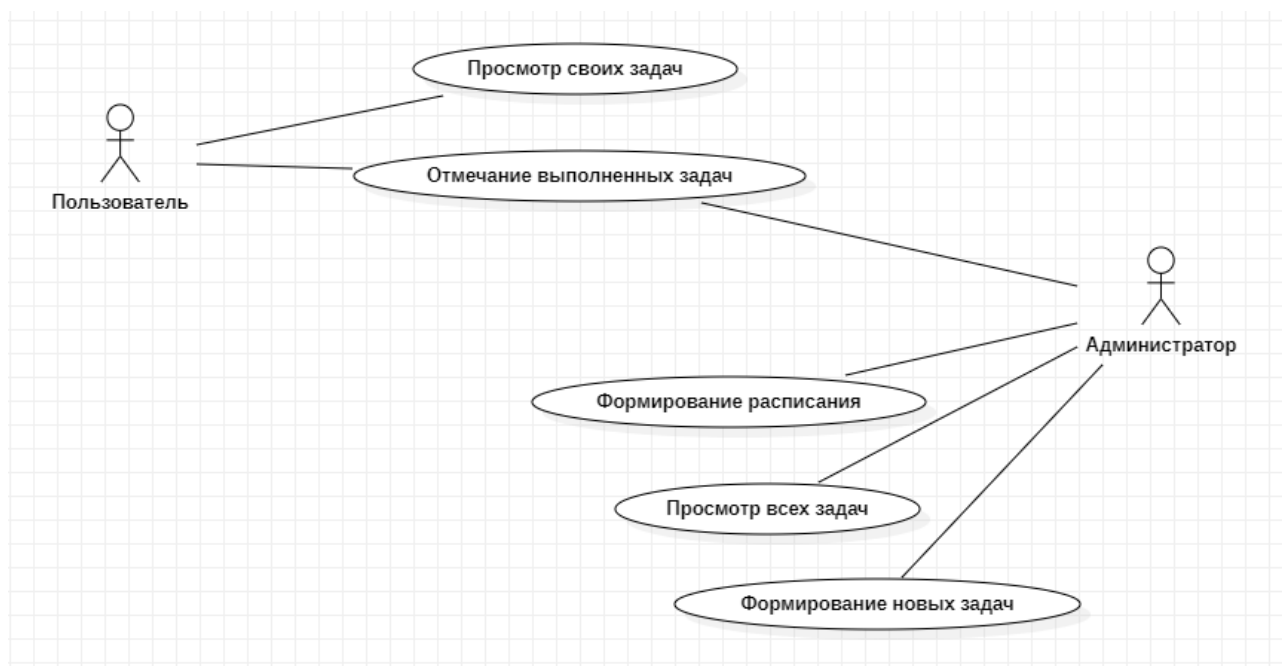


Рисунок 1 – Диаграмма вариантов использования

Описание действующих лиц представлено на таблице 1, а краткое описание вариантов использования на таблице 2. Также в приложении А приведена диаграмма с использованием включений, обобщений.

Таблица 1 – Описание действующих лиц

Название	Профиль, подготовка и навыки
Администратор	Лицо, занимающееся администрированием системы контроля времени. Может вносить изменения в работу приложения
Пользователь	Лицо, имеющее или не имеющее опыт работы с данным приложением, ознакомлен с узким функционалом программы. Вносить изменения и корректировать работу приложения не может

Таблица 2 – Описание вариантов использования

Действующее лицо	Цель	Краткое описание
1	2	3
Администратор	Формирование расписания	Позволяет администратору формировать дневное расписание и распределять задачи между сотрудниками
Администратор	Просмотр всех задач	Позволяет просматривать список всех задач и статус их выполнения

Продолжение таблицы 2

1	2	3
Администратор	Формирование новых задач	Позволяет создавать новые задачи и назначать их сотрудникам
Пользователь	Просмотр своих задач	Позволяет посмотреть список своих задач
Пользователь	Отмечание выполненных задач	Позволяет менять статус своих задач

Более подробно некоторые из прецедентов описаны в диаграммах UML в приложении А.

1.3.2 Список нефункциональных требований

Требования к временным характеристикам:

- время открытия главной страницы не должно быть более 20 с;
- время перехода между страницами не должно быть более 10 с.

1.3.3 Требования к надежности

Администрирование системы.

Доступ к админ-панели может быть получен только администратором путем его перехода к админ-панели по специальной ссылке и авторизации с помощью личного логина и пароля;

Необходимое время для восстановительных мер после отключения.

При отключении сервера, необходимо обеспечивать восстановление информации утраченной в результате сбоя в работе.

1.3.4 Условия эксплуатации

Для успешного функционирования данного клиент-серверного приложения необходима мультипроцессорная операционная система, современный web-браузер, СУБД MySQL, подключение к глобальной сети Интернет. Система должна быть ориентирована на простого пользователя с навыками работы в web-браузерах.

2 Проектирование программного обеспечения

2.1 Проектирование пользовательского интерфейса

Пользовательский интерфейс – это совокупность программных средств, обеспечивающих взаимодействие пользователя с программным продуктом. Основу такого взаимодействия составляют диалоги. Под диалогом в данном случае можно понимать обмен информацией между человеком и программным средством, осуществляемый в реальном масштабе времени и направленный на совместное решение конкретной задачи.

Пользовательский интерфейс информационной системы контроля рабочего времени сотрудников представлен ниже на рисунке 2.

Сотрудники:

1. Иванов Александр Сергеевич

2. Петрова Марина Владимировна

3. Смирнов Дмитрий Алексеевич

4. Кузнецова Анна Михайловна

5. Соколов Иван Викторович

6. Васильева Елена Николаевна

7. Попов Сергей Владимирович

8. Морозова Ольга Петровна

9. Ковалев Андрей Иванович

10. Новикова Наталья Юрьевна

11. Фёдоров Алексей Иванович

12. Белова Екатерина Сергеевна

13. Михайлов Роман Дмитриевич

14. Тихонова Ирина Александровн

15. Орлов Виктор Михайлович

16. Николаева Татьяна Васильевн

+

ред.

Информация о сотруднике:

Соколов Иван Викторович

Обновить

2 мая 2024 г.

Задачи на сегодня:

Задача	Время	Статус	Выполнено
Связаться с клиен...	До 12:00	Не дозвонился	☐
Составить КП	13:00	Составлено	☒
Составить отчет	17:00	Не приступал(а)	☐
**			

Добавить задачу

Редактировать задачу

Рисунок 2 – Стартовая страница системы

Для администратора имеется собственная форма авторизации, позволяющая ему получить доступ к операциям создания/изменения/удаления расписания и задач. Она включает в себя два поля, заполнив которые администратор может получить доступ к админ панели. Зарегистрироваться администратор не может, лишь авторизоваться. Аккаунты администратора создает «супер» администратор через панель управления пользователями. Пользователь может как зарегистрироваться, так и войти в систему. Окно регистрации представлено на рисунке 3.

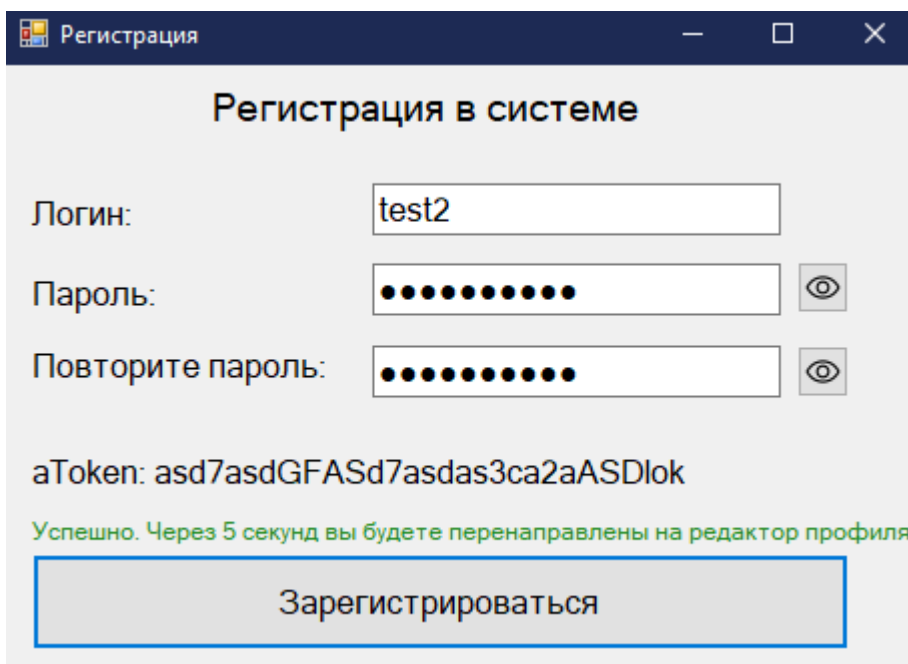
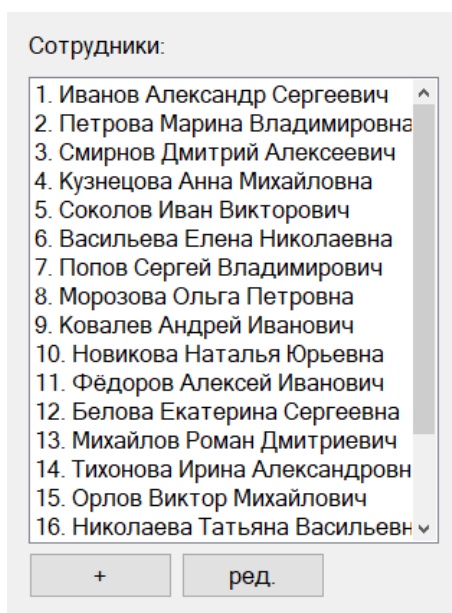


Рисунок 3 – Форма входа администратора

После успешной авторизации администратор попадает в админ панели, внешний вид которой представлен на рисунке 4.



Сотрудники:	
1.	Иванов Александр Сергеевич
2.	Петрова Марина Владимировна
3.	Смирнов Дмитрий Алексеевич
4.	Кузнецова Анна Михайловна
5.	Соколов Иван Викторович
6.	Васильева Елена Николаевна
7.	Попов Сергей Владимирович
8.	Морозова Ольга Петровна
9.	Ковалев Андрей Иванович
10.	Новикова Наталья Юрьевна
11.	Фёдоров Алексей Иванович
12.	Белова Екатерина Сергеевна
13.	Михайлов Роман Дмитриевич
14.	Тихонова Ирина Александровна
15.	Орлов Виктор Михайлович
16.	Николаева Татьяна Васильевна

Рисунок 4 – Интерфейс админ панели

Из данной панели администратор может просматривать список сотрудников, добавлять/изменять/удалять сотрудников, нажимая на

соответствующие кнопки, либо же изменять содержимое задач, путем выбора сотрудников и редактирования задач каждого сотрудника.

2.2 Анализ бизнес процессов организации в нотации IDEF0

IDEF0 – методология функционального моделирования (англ. function modeling) и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является ее акцент на соподчиненность объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность

Бизнес-процесс включает в себя взаимосвязанные действия, которые реализуют одну (или несколько) из бизнес-целей компании в информационной системе компании.

За основу был выбран бизнес-процесс «Контроль рабочего времени», который реализует основную логику проектируемой и разрабатываемой автоматизированной информационной системы.

На контекстной диаграмме организации работы системы контроля рабочего времени сотрудников видно функционирование основной бизнес-функции:

- данные сотрудников (Вход);
- список задач и сроки выполнения (Выход);
- нормативные документы (Управление);
- администратор (Механизм).

Были выделены следующие функциональные процессы: выбор сотрудника, анализ работы сотрудника, формирование задач на месяц, формирование дневного расписания, установка сроков выполнения задач. Таким образом были отображены все функциональные возможности системы.

Диаграмма бизнес-процессов в нотации IDEF0 [3] представлена на рисунке 5.

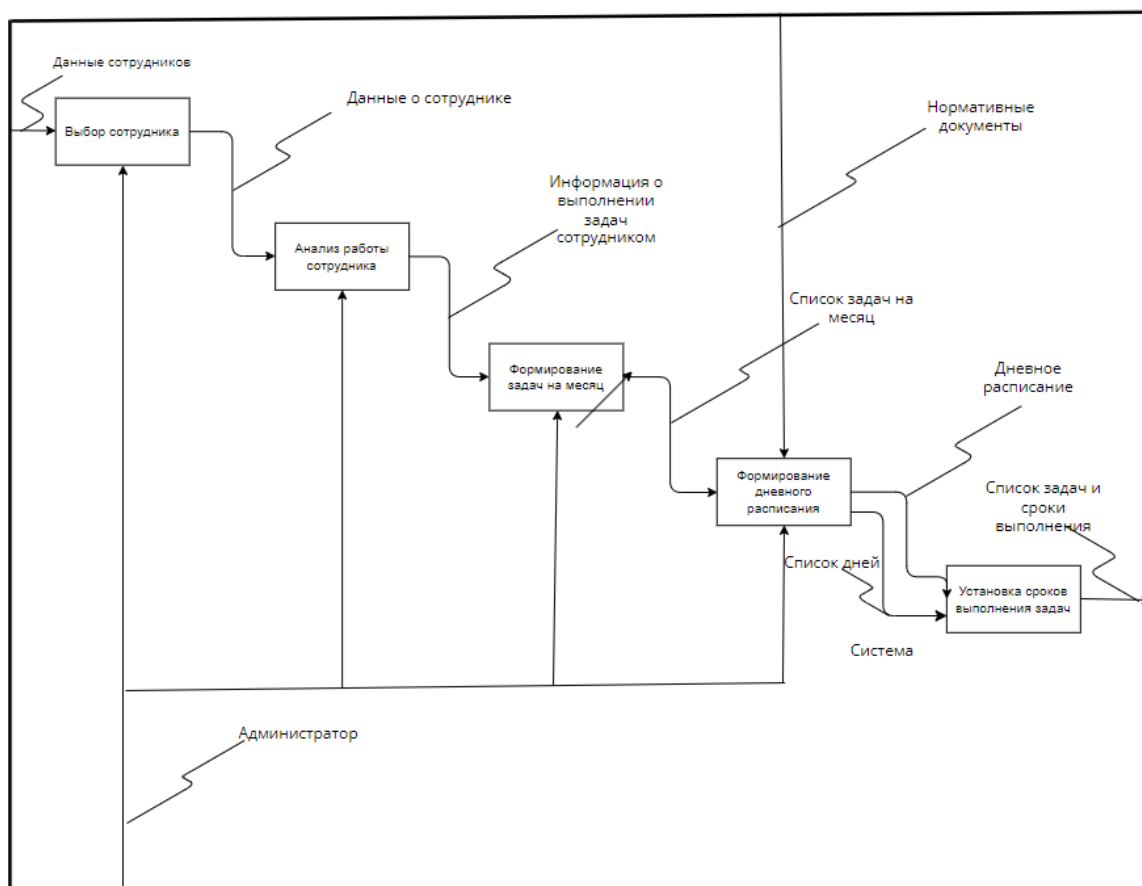


Рисунок 5 – Контекстная диаграмма деятельности по организации работы системы контроля рабочего времени сотрудников

При декомпозиции главной бизнес-функции выделяются следующие функциональные блоки: выбор сотрудника, анализ работы сотрудника, формирование задач на месяц, формирование дневного расписания, установка сроков выполнения задач.

2.3 Проектирование модели базы данных

Для разработки моделей базы данных используется Case-средство ERwin и его методологию IDEF1X.

В ERwin существуют два уровня представления и моделирования – инфологический и физический.

2.3.1 Проектирование физической и инфологической моделей базы данных с помощью Erwin

Инфологический уровень – это общий взгляд на данные, на нем данные представляются так, как выглядят в реальном мире, и могут называться так, как они называются в реальном мире. Объекты модели, представляемые на инфологическом уровне, называются сущностями и атрибутами. Инфологическая модель данных является универсальной и никак не связана с конкретной реализацией СУБД.

Создание модели данных, как правило, начинается с создания инфологической модели. После описания инфологической модели, человек, занимающийся проектированием может выбрать необходимую СУБД и ERwin [4] автоматически создаст соответствующую физическую модель. На основе физической модели ERwin может сгенерировать системный каталог СУБД или соответствующий SQL-скрипт [5], необходимый для создания базы данных.

В Case средстве ERwin [6] разработана инфологическая модель базы данных для хранения информации организации с учетом всех нюансов и специфики данной предметной области. Данная модель содержит восемь таблиц, связанных между собой логическими связями.

Были определены следующие таблицы: расписание (для хранения информации о расписании), ответственный (ответственное лицо, назначенное за контролем), администратор (администраторы системы), строка (одна запись в расписании), задача (целая задача, назначенная сотруднику или группе сотрудников), группа сотрудников (список сотрудников, выделенных в отдельную категорию), сотрудники (информация об одном сотруднике), отдел (информация об отделе, к которому привязана группа сотрудников или один сотрудник отдельно).

Инфологическая модель представлена на рисунке 6.

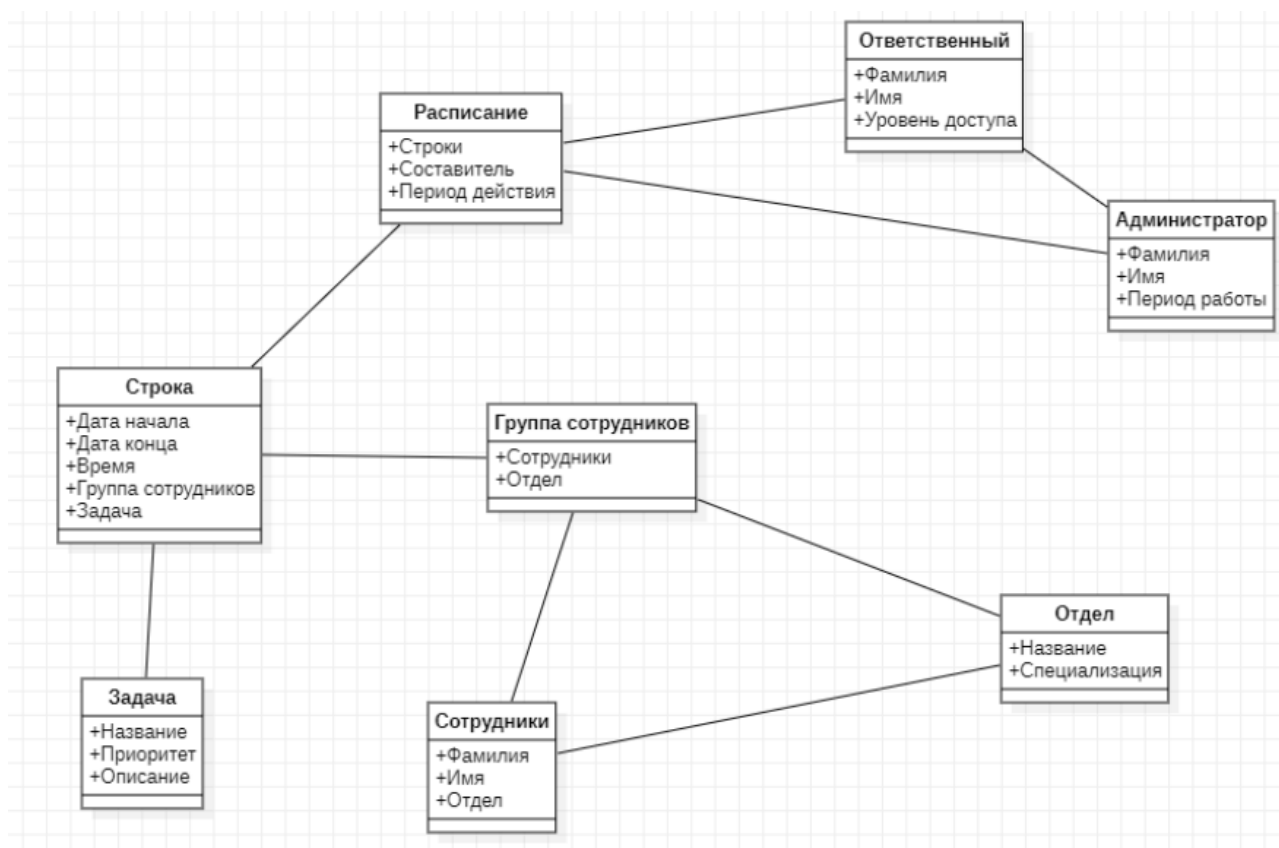


Рисунок 6 – Диаграмма инфологической модели

Физическая модель данных зависит от конкретной СУБД, фактически являясь отображением системного каталога. В физической модели содержится информация о всех объектах БД. Поскольку стандартов на объекты БД не существует (например, нет стандарта на типы данных), физическая модель зависит от конкретной реализации СУБД. Следовательно, одной и той же инфологической модели могут соответствовать несколько разных физических моделей (например, для СУБД MySQL и СУБД MySQL физические модели будут отличаться).

В данном случае была выбрана СУБД MySQL, физическая модель была построена для выбранной СУБД. Таким образом были указаны типы данных для атрибутов модели каждой таблицы.

Физическая модель представлена на рисунке 7.

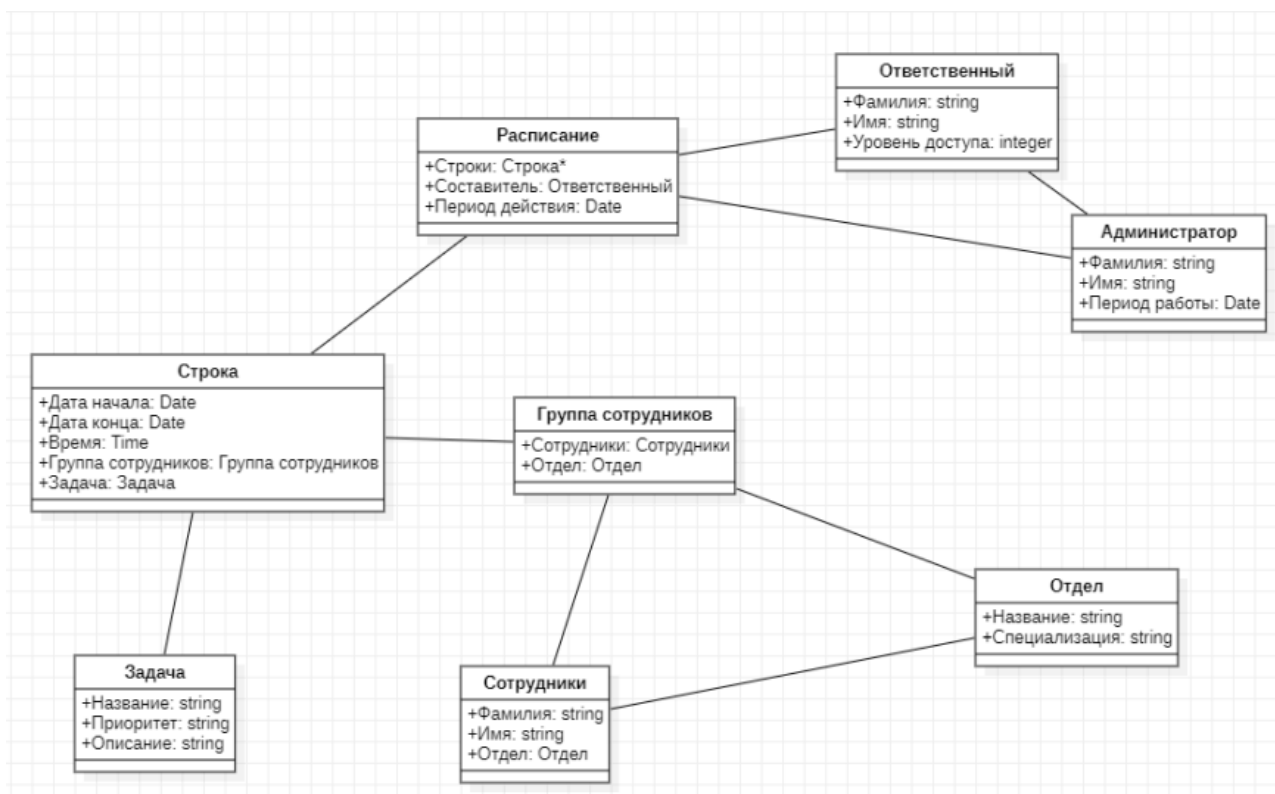


Рисунок 7 – Диаграмма физической модели

Если в инфологической модели не имеет значения, какой конкретно тип данных имеет атрибут, то в физической модели дело обстоит иначе – важно описать всю информацию о конкретных физических объектах, таких как таблицы, колонки, индексы, первичные и внешние ключи.

На диаграмме IDEF1X [7] физического уровня представлены таблицы и связи между ними, которые будут реализованы в разрабатываемой базе данных.

2.4 Проектирование поведения ИС

Для отражения взаимодействия информационной системы контроля рабочего времени и ее компонентов построены поведенческие диаграммы: диаграмма состояния, диаграмма деятельности, диаграмма последовательности, диаграмма коммуникации, диаграмма компонентов и диаграмма развертывания.

2.4.1 Диаграмма состояний

Объекты характеризуются поведением и состоянием, в котором находятся. Диаграмма состояний показывает, как объект переходит из одного состояния в другое. В данном случае диаграмма состояний отражает состояние объекта класса «Расписание», и его возможные состояния.

Класс «Расписание», помимо начального и конечного состояния может, находиться в следующих состояниях:

- `rasp is Empty` (расписание пустое) – в начале дня, расписание еще может быть не заполнено;
- `rasp contains tasks` (расписание содержит задачи на день) – расписание может содержать задачи, после заполнения их администратором в день расписания или заранее.

Переходы в указанные состояния на диаграмме имеют сторожевые условия (условия перехода).

2.4.2 Диаграмма деятельности

Диаграмма деятельности показывает разложение некоторой деятельности на ее составные части. Под деятельностью понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчиненных элементов – вложенных видов деятельности и отдельных действий, соединенных между собой потоками, которые идут от выходов одного узла ко входам другого.

Диаграмма деятельности построена для прецедента «Создать задачу».

На диаграмме деятельности отражено выполнение действий. Сначала необходимо выбрать сотрудника. Если у сотрудника еще есть время на выполнение новых задач, открывается форма заполнения данных о новой задаче. Администратор заполняет ее и нажимает кнопку «Сохранить».

Диаграмма представлена в приложении А.

2.4.3 Диаграмма последовательности

Диаграмма последовательности – диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл какого-либо определенного объекта и взаимодействие актеров ИС в рамках какого-либо определенного прецедента (отправка запросов и получение ответов). Для разрабатываемой ИС диаграмма последовательности построена для прецедента «Добавление нового сотрудника». На ней «Пользователь» актер взаимодействует с объектами системы – Формой заполнения данных, Модулем работы с данными и базой данных – отправляя и получая сообщения.

Диаграмма представлена в приложении А.

2.4.4 Диаграмма коммуникаций

Диаграмма коммуникаций – диаграмма, на которой изображаются взаимодействия между частями композитной структуры или ролями кооперации. В отличие от диаграммы последовательности, на диаграмме кооперации явно указываются отношения между объектами, а время как отдельное измерение не используется (применяются порядковые номера вызовов). Для информационной системы контроля рабочего времени сотрудников построена диаграмма коопераций для процесса «администрирование расписания».

На диаграмме коммуникаций представлены следующие объекты:

- администратор;
- система;
- админ панель;
- база данных.

Администратор переходит в админ панель, система возвращает ему форму авторизации, администратор вводит свои логин и пароль. После этого она попадает в админ-панель, где создает/изменяет/удаляет задачи, и

отправляет запрос в БД. После успешного проведения операции БД возвращает измененную таблицу в админ панель.

Диаграмма представлена в приложении А.

2.4.5 Диаграмма компонентов

Диаграмма компонентов – статическая структурная диаграмма, показывает разбиение программной системы на структурные компоненты [8] и связи (зависимости) между компонентами. В качестве физических компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты.

Автоматизированная информационная система контроля рабочего времени содержит в себе следующие структурные компоненты:

- loginViewModel.cs, RaspIndexModel.cs, PagingInfo.cs, TasksListViewModel.cs – модели;
- login.php, admin.Edit.php, rasp.checktask.php, rasp.compltask.php, rasp.Index.php, rasp.unvtasks.php – представления;
- accountController.cs, AdminController.cs, TaskController.cs, NavController.cs, RaspController.cs – контролеры.

Компоненты связаны между собой отношениями зависимости.

Диаграмма представлена в приложении А.

2.4.6 Диаграмма развертывания

Диаграмма развертывания – это диаграмма, на которой представлены узлы выполнения программных компонентов реального времени, а также процессов и объектов.

Диаграмма развертывания применяется для представления общей конфигурации и топологии распределенной программной системы и содержит изображение размещения компонентов по отдельным узлам системы. Кроме того, диаграмма развертывания показывает наличие физических соединений

маршрутов передачи информации между аппаратными устройствами, задействованными в реализации системы.

Диаграмма развертывания предназначена для визуализации элементов компонентов программы, существующих только на этапе ее исполнения. Выделим основные компоненты web-приложения информационной системы контроля рабочего времени сотрудников. В компоненте MySQL DB находится база данных нашей информационной системы. На компоненте ISS 11 web-сервер находится начальная страница web-приложения ИС <http://localhost/port>. Компонент браузер служит для подключения к стартовой странице приложения.

Компоненты связаны между собой отношениями зависимости и интерфейсами.

Диаграмма представлена в приложении А.

3 Разработка приложения

3.1 Программные средства

После ознакомления с основными существующими на данный момент типами баз данных, выбор пал на реляционную базу данных на основе MySQL 2022. Microsoft MySQL – система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основным используемый язык запросов – Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями.

SQL (Structured Query Language) [9] – стандартный язык запросов по работе с реляционными базами данных. SQL в чистом (базовом) виде является информационно-логическим языком, а не языком программирования. Однако стандарт языка спецификацией SQL/PSM предусматривает возможность его процедурных расширений, с учетом которых язык уже может рассматриваться в качестве языка программирования. SQL предоставляет пользователю достаточно простой и понятный механизм доступа к данным, не связанный с конструированием алгоритма и его описанием на языке программирования высокого уровня. Так, вместо указания того, как необходимо действовать, пользователь с помощью операторов SQL объясняет СУБД, что нужно сделать. Далее СУБД сама анализирует текст запроса и определяет, как именно его выполнять, и после этого выдает результат запроса.

Соединение с базой данных [10] и клиентского приложения осуществляется с помощью языковых средств C# [11, 12], а именно его фреймворка ASP.NET в среде разработки Visual Studio 2022.

C# – это язык программирования, предназначенный для разработки самых разнообразных приложений, предназначенных для выполнения в среде .NET Framework. Язык C# прост, безопасен и объектно-ориентирован. Благодаря множеству нововведений C# обеспечивает возможность быстрой

разработки приложений, но при этом сохраняет выразительность и элегантность, присущую языкам С. Так же весомым преимуществом языка С# является то, что он постоянно обновляется, благодаря чему пользователю не приходится прибегать к сторонним средствам для выполнения поставленных задач. Достаточно загрузить нужные библиотеки, добавляющие необходимый функционал.

Платформа ASP.NET MVC [13,14] представляет технологию от компании Microsoft, предназначенную для создания различного рода Web-приложений: от небольших web-сайтов до крупных web-порталов и web-сервисов.

С одной стороны, ASP.NET MVC является продолжением развития платформы ASP.NET. Но с другой стороны, это не просто очередной релиз. Выход ASP.NET MVC фактически означает революцию всей платформы, ее качественное изменение.

С помощью ASP.NET MVC разработчик может создавать кроссплатформенные приложения. И хотя Windows в качестве среды для разработки и развертывания приложения до сих пор превалирует, но теперь он уже не ограничен только этой операционной системой. То есть разработчик может запускать web-приложения не только на ОС Windows, но и на Linux и Mac OS. А для развертывания web-приложения можно использовать традиционный IIS, либо кросс-платформенный web-сервер

Microsoft Visual Studio – линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии как устаревшей технологии Windows Forms, так и более новой Windows Presentation Foundation. Так же эти продукты позволяют разрабатывать веб-сайты, Web-

приложения, web-службы как в родном, так и в управляемом кодах для всех платформ.

Преимущества ASP.NET MVC:

- архитектурный шаблон MVC;
- расширяемость;
- жесткий контроль над HTML и HTTP;
- тестируемость.

В результате, с использованием данных средств приложение было реализовано и запущено. Текст приложения приведен в приложении Б.

4 Отладка и тестирование

4.1 Тестирование программного обеспечения

Тестирование [15] – очень важный этап разработки как приложений. По способам, целям и глубине тестирования оно делится на следующие виды:

- функциональное тестирование (functional testing) [16];
- системное тестирование (system testing);
- тестирование производительности (performance testing);
- регрессионное тестирование (regression testing);
- модульное тестирование (unit testing);
- тестирование безопасности (security testing);
- тестирование локализации (localization testing);
- юзабилити тестирование (usability testing).

Модульное тестирование (Unit Testing) – это тип тестирования программного обеспечения, при котором тестируются отдельные модули или компоненты программного обеспечения. Его цель заключается в том, чтобы проверить, что каждая единица программного кода работает должным образом. Данный вид тестирования выполняется разработчиками на этапе кодирования приложения. Модульные тесты изолируют часть кода и проверяют его работоспособность. Единицей для измерения может служить отдельная функция, метод, процедура, модуль или объект. Отсутствие модульного тестирования при написании кода значительно увеличивает уровень дефектов при дальнейшем (интеграционном, системном, и приемочном) тестировании. Качественное модульное тестирование на этапе разработки экономит время, а следовательно, в конечном итоге, и бюджет проекта.

Проведем модульное тестирование [17] работы отдельных функций приложения. Таблица с описанием тестов представлена ниже (таблица 3).

Результаты тестирования приведены в приложении В.

Таблица 3 – Модульное тестирование работы системы

№	Название теста	Предполагаемый результат	Полученный результат	Результат тестирования
1	2	3	4	5
1	Adding_Record_To_Rasp_Goes_To_Rasp_Screen	При добавлении записи в форму открывается модуль	Соответствует ожидаемым результатам	Тест пройден
2	Calculate_Rasp_Total	Подсчет итоговой загруженности дня	Соответствует ожидаемым результатам	Тест пройден
3	Can_Add_New_Lines	Добавление новых строк в базу данных из админ-панели работает верно	Соответствует ожидаемым результатам	Тест пройден
4	Can_Add_Quantity_For_Existing_Lines	Повторное добавление записи приводит к увеличению счетчика	Соответствует ожидаемым результатам	Тест пройден
5	Can_Add_To_Rasp	Добавление записи в базу данных работает верно	Соответствует ожидаемым результатам	Тест пройден
6	Can_Checkout_And_Submit_Order	Формирование и отправка запроса работает верно	Соответствует ожидаемым результатам	Тест пройден
7	Can_Clear_Contents	Функция очистки дня работает верно	Соответствует ожидаемым результатам	Тест пройден
8	Can_Create_Users	Добавление новых пользователей работает верно	Соответствует ожидаемым результатам	Тест пройден

Продолжение таблицы 3

1	2	3	4	5
9	Can_Delete_Valid_Records	Удаление записей из админ панели работает верно	Соответствует ожидаемым результатам	Тест пройден
10	Can_Edit_Record	Изменение записей из админ панели работает верно	Соответствует ожидаемым результатам	Тест пройден
11	Can_Generate_Page_Links	Генерация ссылок на страницы работает верно	Соответствует ожидаемым результатам	Тест пройден
12	Can_Login_With_Valid_Credits	Авторизация администратора для доступа к админ-панели проходит успешно	Соответствует ожидаемым результатам	Тест пройден
13	Can_Paginate	Разбиение на страницы работает верно	Соответствует ожидаемым результатам	Тест пройден
14	Can_Remove_Line	Удаление отдельных записей из дня работает верно	Соответствует ожидаемым результатам	Тест пройден
15	Can_Retrieve_Info_Data	Добавление информации пользователей работает верно	Соответствует ожидаемым результатам	Тест пройден
16	Can_Save_Valid_Changes	Изменения, соответствующие требованиям, успешно сохраняются	Соответствует ожидаемым результатам	Тест пройден

Окончание таблицы 3

1	2	3	4	5
17	Can_View_Rasp_Contents	Пользователь может просматривать расписание	Соответствует ожидаемым результатам	Тест пройден
18	Cannot_Checkout_Empty_Rasp	Невозможно отправить заявку при пустой записи	Соответствует ожидаемым результатам	Тест пройден
19	Cannot_Edit_Nonexist_Record	Невозможно редактировать несуществующую запись	Соответствует ожидаемым результатам	Тест пройден
20	Cannot_Login_Without_Valid	Невозможно войти по не верным данным аккаунта	Соответствует ожидаемым результатам	Тест пройден
21	Cannot_Retrive_Info_Data	Невозможно присвоить сведения несуществующей записи	Соответствует ожидаемым результатам	Тест пройден
22	Cannot_Save_Invalid_Changes	Невозможно сохранить изменения, если они не соответствуют требования	Соответствует ожидаемым результатам	Тест пройден
23	Generate_Day_Specific_Records	Количество записей подсчитывается верно	Соответствует ожидаемым результатам	Тест пройден
24	Index_Contains_All_Records	Расписание содержит все записи, содержащиеся в базе данных	Соответствует ожидаемым результатам	Тест пройден

Модульное тестирование [18] показало, что система функционирует корректно и не имеет явных ошибок и сбоев в работе, что является положительным результатом проведения тестирования.

4.1.1 Метрики программного средства

Метрика программного обеспечения – мера, позволяющая получить численное значение некоторого свойства программного обеспечения или его спецификаций.

Метрики используются в следующих целях:

- для контроля над процессом разработки в целом, а в тестировании в частности;
- для оценки прогресса выполнения работ по тестированию ПО;
- для последующего планирования процессов.

4.1.2 Расчет метрик кода с помощью встроенного средства Visual Studio

Среда разработки Microsoft Visual Studio 2022 имеет встроенное средство для выполнения автоматического расчета метрик кода. К таковым относятся:

- индекс удобства поддержки;
- сложность организации циклов;
- глубина наследования;
- объединение классов;
- строки кода.

Для расчета метрик кода средствами Microsoft Visual Studio 2022 необходимо выбрать пункт контекстного меню «Анализ», далее – «Рассчитать метрики кода».

На рисунке 8 приведен результат расчета метрик кода в среде Microsoft Visual Studio 2022.

Компонент	Индекс удобства поддержки	Сложность организации циклов	Глубина наследования	Взаимозависимость классов	Строки кода
nain (Debug)	84	71	2	32	108
Domain.Abstract	100	4	0	4	0
Domain.Concrete	72	21	2	20	54
Domain.Entities	89	46	1	16	54
ITests (Debug)	67	47	1	76	245
UnitTests	64	38	1	62	232
UnitTests.App_Start	81	9	1	17	13
bUI (Debug)	80	102	3	99	193
WebUI	80	4	2	16	11
WebUI.App_Start	80	9	1	19	14
WebUI.Controllers	71	47	3	43	104
WebUI.HtmlHelpers	59	3	1	7	12
WebUI.Infrastructure	80	5	1	19	11
WebUI.Infrastructure.	100	1	0	0	0
WebUI.Infrastructure.I	71	5	1	6	10
WebUI.Infrastructure.I	80	3	1	2	6
WebUI.Models	93	25	1	7	25

Рисунок 8 – Метрики кода в Visual Studio

Для наглядности среда Microsoft Visual Studio отображает цветовую индикацию уровня качества кода. Зеленая цвет свидетельствует о высоком качестве кода, желтая означает, что качество кода ниже оптимального. В данном случае видно, что качество кода всех компонентов имеет высокое качество.

4.1.3 Определение метрик компонента серверной части приложения

Далее рассмотрен исходный код метода Login класса AccountController.cs (рисунок 9), для которого будут рассчитаны метрики Холстеда и выполнен комплексный анализ кода. Анализ кода будет выполнен согласно формулам метрики Холстеда.

```
public ActionResult Login(LoginViewModel model, string returnUrl)
{
    if (ModelState.IsValid)
    {
        if (authProvider.Authenticate(model.UserName, model.Password))
        {
            return Redirect(returnUrl ?? Url.Action("Index", "Admin"));
        }
        else
        {
            ModelState.AddModelError("", "Неправильный логин или пароль");
            return View();
        }
    }
    else
    {
        return View();
    }
}
```

Рисунок 9 – Код метода Login(LoginViewModel, string)

Словарь для расчета метрик вышеуказанного блока исходного кода приведен в таблице 2.

Таблица 4 – Словарь

Переменная	№	Встречается	Оператор	№	Встречается
model	1	1	if	1	2
returnUrl	2	1	return	2	3
ModelState	3	2	else	3	2
authProvider	4	1	??	4	1
UserName	5	1			
Password	6	1			
Итого:	6	7		4	8
	n1	N1		n2	N2

Расчет метрик Холстеда выполнен с использованием математического пакета Mathcad. Результат расчета приведен ниже.

Исходные данные:

$$mn1 := 6, mN1 := 7, mn2 := 4, mN2 := 8.$$

Словарь, длина и объем программы:

$$n := mn1 + mn2 = 10;$$

$$N := mN1 + mN2 = 15;$$

$$V := N * \log(2, n) = 49,83.$$

Уровень качества программирования:

$$L := 2mn2 / (mn1 * mN2) = 0,17.$$

Сложность понимания программы.

$$EC := V/2L = 146,55.$$

Уровень языка:

$$Y := V/D2 = 1,38.$$

Оценка необходимых интеллектуальных усилий:

$$E := N * \log(2, n/L) = 87,87.$$

Трудоемкость кодирования:

$$D := 1/L = 6.$$

Информационное содержание программы:

$$I := V/D = 8,31.$$

Подход, приведенный в расчете метрик Холстеда, значительно усложняется при увеличении объема кода проекта. В таком случае целесообразно использовать встроенные в среду разработки или отдельно поставляемые инструменты для автоматического анализа качества кода.

Заключение

С помощью языка C#, базы данных MySQL 2022 и среды разработки Visual Studio 2022 от компании Microsoft были реализованы база данных и клиент-серверное приложение информационной системы контроля рабочего времени сотрудников. По результатам функционального тестирования автоматизированная информационная система работает корректно и выполняет поставленные задачи.

Применение данной системы позволит организовать деятельность любой компании, где требуется вести учет времени сотрудников:

- быстро и удобно вести администрирование имеющихся задач сотрудников;
- снизить случайные ошибки связанных с ручным оформлением задач;
- автоматизировать обработку информации в системе;
- получить доступ к списку задач из любой точки, где есть компьютер, имеющий выход в Интернет;
- получить на выходе систему с низкой стоимостью разработки, но имеющую весь основной функционал, необходимый для комфортного и удобного контроля за работой сотрудников.

Список использованных источников

1. Информационная система – это //Что такое информационные системы и зачем они нужны – классификация и использование [Электронный ресурс] URL: <https://ktonanovenkogo.ru/voprosy-i-otvety/informacionnayasisistema-chto-eh-to-takoe.html?ysclid=l4hevni6mn440545180> (дата обращения 07.05.2024).
2. Вендров А.М. Структурный подход к проектированию ИС // Портал научно-практических публикаций [Электронный ресурс]. URL: http://citforum.ru/database/case/glava2_1.shtml (дата обращения: 07.05.2024).
3. Приложение с открытым исходным кодом для персональных компьютеров // ArgoUML [Электронный ресурс]. URL: <https://argouml.softonic.ru> (Дата обращения: 15.05.2024).
4. Построение UML-модели в среде Visual Paradigm 16.1. Система обработки заказов [Электронный ресурс]. URL: <http://sp.cs.msu.ru/ooap/exerb2020.html> (Дата обращения: 18.05.2024).
5. Маклаков, С. В. BPwin и ERwin: CASE-средства для разработки информационных систем // Портал научно-практических публикаций [Электронный ресурс]. URL: <https://elib.pnzgu.ru/files/eb/doc/DSA5gziAMfNS.pdf> (дата обращения: 18.05.2024).
6. Проектирование информационных систем [Электронный ресурс]. URL: <https://moodle.kstu.ru/course/view.php?id=4638&> (Дата обращения: 26.05.2024).
7. Варианты использования // Диаграмма вариантов использования [Электронный ресурс]. URL: http://www.telenir.net/uchebniki/samouchitel_uml/p4.php?ysclid=l4hg07qpf139464158 (Дата обращения: 28.05.2024).
8. Варианты использования // Диаграмма вариантов использования [Электронный ресурс]. URL:

http://www.telenir.net/uchebniki/samouchitel_uml/p4.php?ysclid=l4hg07qpf139464158 (Дата обращения: 28.05.2024).

9. Что такое интерфейс пользователя? [Электронный ресурс]. URL: <https://www.internet-technologies.ru/articles/newbie/polzovatelskiy-interfeys.html> (Дата обращения: 01.06.2024).

10. Простое руководство по диаграмме компонентов: [Электронный ресурс] // So creately URL: <https://creately.com/blog/ru/uncategorized-ru/учебное-пособие-по-компонентной-диаг/>. (Дата обращения: 04.06.2024).

11. Об одном подходе к разбиению на классы эквивалентности: [Электронный ресурс] // cyberleninka.ru. Теория. URL: <https://cyberleninka.ru/article/n/ob-odnom-podhode-k-razbieniyu-na-klassy-ekvivalentnosti> (дата обращения: 26.03.2024).

12. Граничные условия в неизолированных системах: [Электронный ресурс] // cyberleninka.ru. Теория. URL: <https://cyberleninka.ru/article/n/granichnye-usloviya-v-neizolirovannyh-sistemah>. (Дата обращения: 26.03.2024).

13. Простое сложное программирование: [Электронный ресурс] // <http://habr.com>. Теория. URL: <https://habr.com/ru/companies/timeweb/articles/551754/>. (Дата обращения: 26.03.2024).

14. Применение метрик Холстеда для количественного оценивания характеристик программ ЭВМ: [Электронный ресурс] // cyberleninka.ru. Теория. URL: <https://cyberleninka.ru/article/n/primenenie-metrik-holsteda-dlya-kolichestvennogo-otsenivaniya-harakteristik-programm-evm>. (Дата обращения: 26.03.2024).

Приложение А
(рекомендуемое)
Диаграммы UML



Рисунок А.1 – Диаграмма вариантов использования

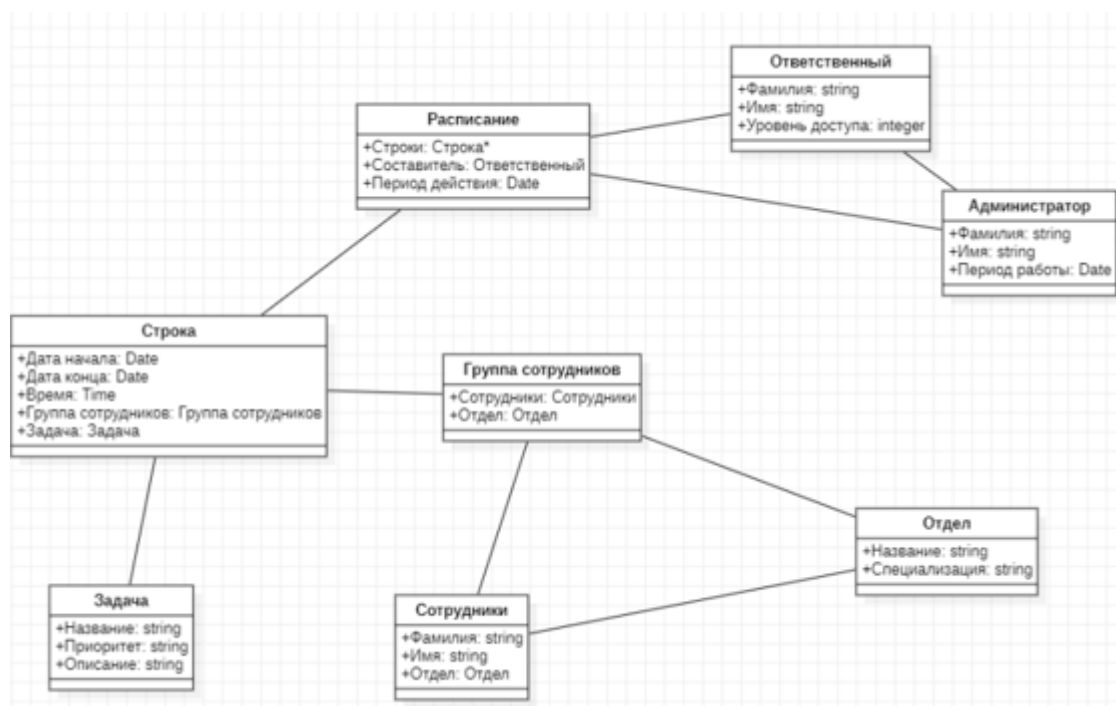


Рисунок А.2 – Диаграмма классов



Рисунок А.3 - Диаграмма коммуникаций

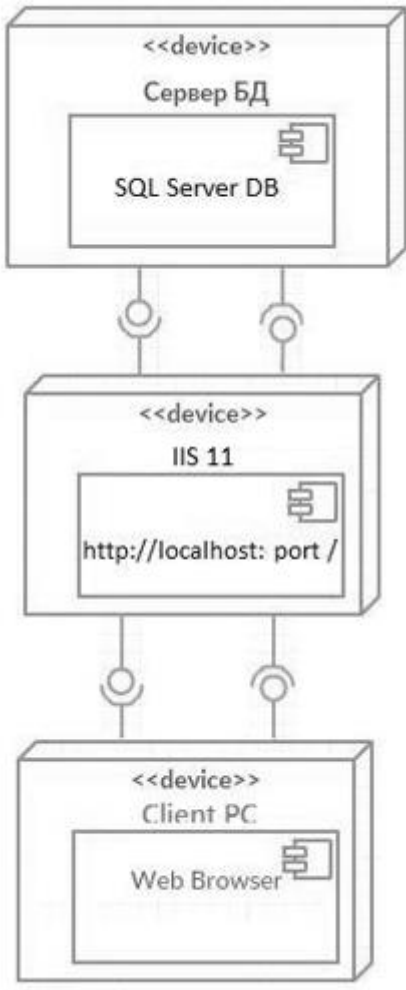


Рисунок А.4 - Диаграмма развертывания

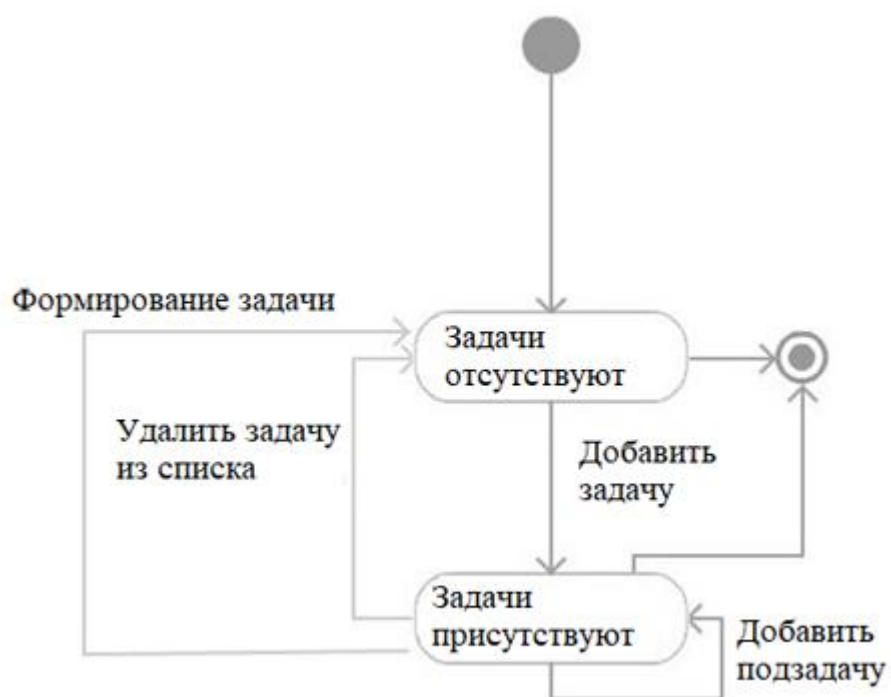


Рисунок А.5 – Диаграмма состояний

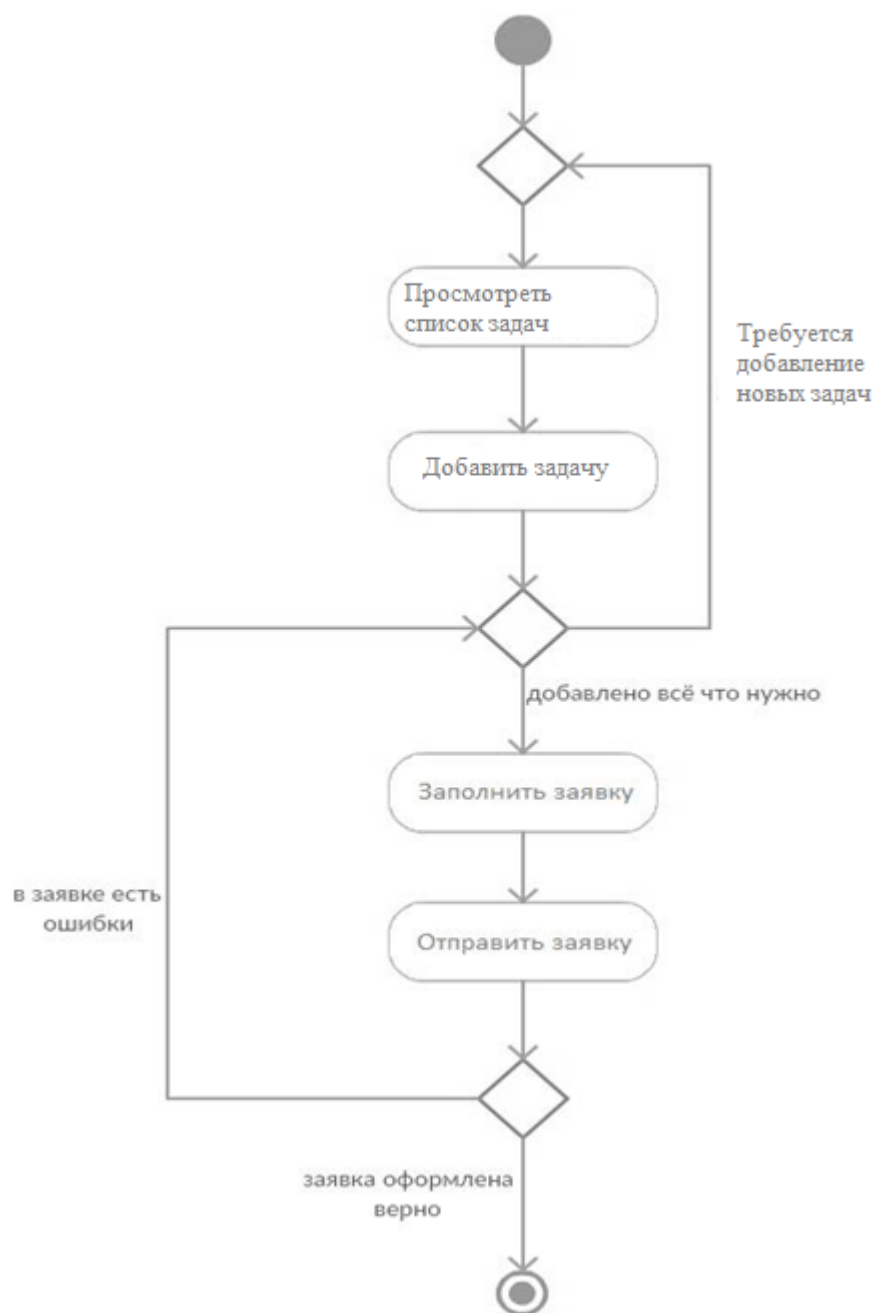


Рисунок А.6 – Диаграмма деятельности

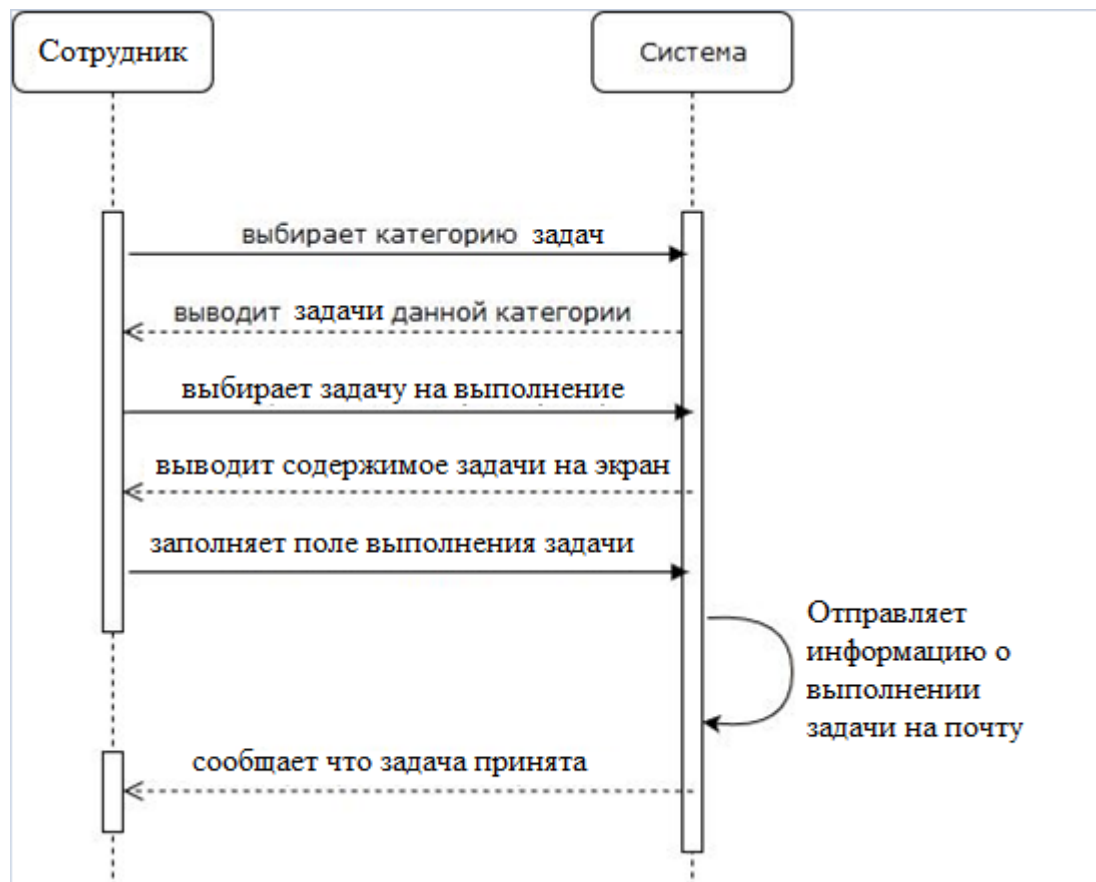


Рисунок А.7 – Диаграмма последовательности

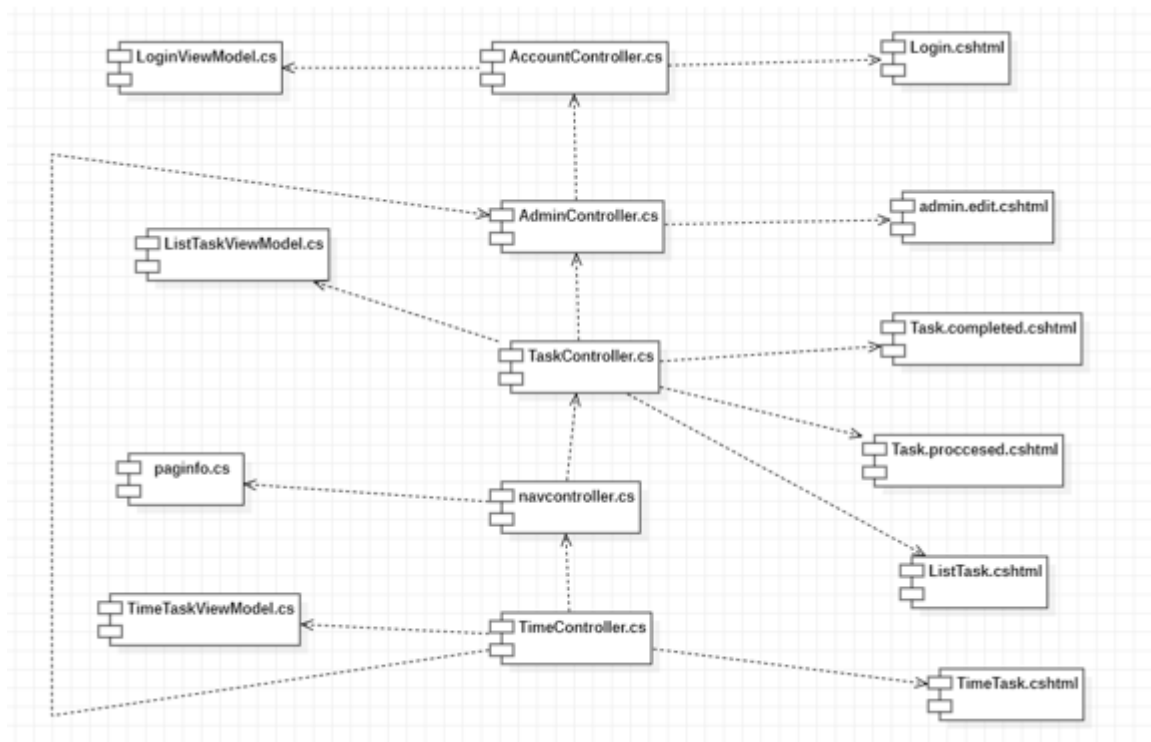


Рисунок А.8 – Диаграмма компонентов

Приложение Б
(рекомендуемое)
Листинг программы

Листинг Б.1 – Код модуля «Rasp.Domain.Concrete.EFDbContext.cs»

```

using System;
using System.Collections.Generic;
using System.Linq;

using System.Text;
using System.Threading.Tasks;
using Rasp.Domain.Entities;

namespace Rasp.Domain.Concrete
{

public class EFDbContext : DbContext
{
public DbSet<Task> Tasks { get; set; }

}

}

```

Листинг Б.2 – Код модуля «Rasp.Domain.ConcreteEFRasp.cs»

```

using System.Collections.Generic;

using Rasp.Domain.Entities;
using Rasp.Domain.Abstract;
using System.Linq;
using System.Text;

namespace Rasp.Domain.Concrete
{

public class EFRaspRepository : IRaspRepository
{
EFDbContext ccontext = new EFDbContext();

public IEnumerable<Task> Task
{

get { return ccontext.Tasks; }

}

public void SaveTask(Task task)

{
if (task.TaskId == 0)
ccontext.Tasks.Add(task);

else
{

```

```

Task dbEntry = ccontext.Tasks.Find(task.TaskId);
if (dbEntry != null)

{
    dbEntry.Name          =      task.Name;          dbEntry.Description      =
task.Description;  dbEntry.Price  =  task.Price;  dbEntry.Category  =
task.Category; dbEntry.ImageData = task.ImageData;

    dbEntry.ImageMimeType = task.ImageMimeType;
}
}

ccontext.SaveChanges();

public Task DeleteTask(int taskId)

{
    Task dbEntry = ccontext.Tasks.Find(taskId);
    if (dbEntry != null)

    {
        ccontext.Tasks.Remove(dbEntry);
        ccontext.SaveChanges();
    }

    return dbEntry;
}
}
}

```

Листинг Б.3 – Код модуля «Rasp.UnitTests. AdminSecurityTests.cs»

```

using System.Web.Mvc;

using Microsoft.VisualStudio.TestTools.UnitTesting; using Moq;
using Rasp.WebUI.Controllers;

using          Rasp.WebUI.Infrastructure.Abstract;          using
Rasp.WebUI.Models; using System.Linq;
using System.Text;

namespace Rasp.UnitTests
{
    [TestClass]
    public class AdminSecurityTests
    {
        [TestMethod]

```

```

    public void Can_Login_With_Valid_Credentials()
    {
//        Организация - создание имитации поставщика аутентификации

        Mock<IAuthProvider>newmock = new Mock<IAuthProvider>();
newmock.Setup(m => m.Authenticate("admin",
    "12345")).Returns(true);

//        Организация - создание модели представления
//        с правильными учетными данными
LoginViewModel model = new LoginViewModel
    {

        UserName = "admin", Password = "12345" };

//        Организация - создание контроллера
AccountController target = new
AccountController(newmock.Object);

//        Действие - аутентификация
ActionResult result = target.Login(model, "/MyURL");

//        Утверждение

        Assert.IsInstanceOfType(result, typeof(RedirectResult));
Assert.AreEqual("/MyURL", ((RedirectResult)result).Url); }

[TestMethod]
public void Cannot_Login_With_Invalid_Credentials()
    {
//        Организация - создание имитации поставщика аутентификации

        Mock<IAuthProvider>newmock = new Mock<IAuthProvider>();
newmock.Setup(m => m.Authenticate("badUser",
    "badPass")).Returns(false);

//        Организация - создание модели представления

//        с некорректными учетными данными
LoginViewModel model = new LoginViewModel
    {

        UserName = "badUser", Password = "badPass" };

//        Организация - создание контроллера
AccountController target = new
AccountController(newmock.Object);

//        Действие - аутентификация
ActionResult result = target.Login(model, "/MyURL");

//        Утверждение

```

```

        Assert.IsInstanceOfType(result,          typeof(ViewResult));
Assert.IsFalse(((ViewResult)result).ViewData.ModelState.IsVa
lid);

    }
}
}

```

Листинг Б.4 – Код модуля Rasp.UnitTests. AdminTests.cs

```

using Rasp.WebUI.Controllers;

using Microsoft.VisualStudio.TestTools.UnitTesting; using Moq;

using Rasp.Domain.Abstract;
using Rasp.Domain.Entities;
using System;

using System.Collections.Generic;

using System.Linq;
using System.Web.Mvc;
using System.Text;

namespace Rasp.UnitTests
{
    [TestClass]

    public class AdminTests
    {
        [TestMethod]

        public void Index_Contains_All_Taskss()
        {
            //      Организация - созданиеимитированногохранилищаданных

            Mock<IRaspRepository>newmock    =    new    Mock<IRaspRepository>();
newmock.Setup(m => m.Task).Returns(new List<Task>
{

    new Task { TaskId = 1, Name = "Задача1"}, new Task { TaskId =
2, Name = "Задача2"}, new Task { TaskId = 3, Name = "Задача3"},
new Task { TaskId = 4, Name = "Задача4"}, new Task { TaskId = 5,
Name = "Задача5"} });

            //      Организация - созданиеконтроллера

            AdminController          controller          =          new
AdminController(newmock.Object);

            //      Действие

```

```

        List<Task> result =
            ((IEnumerable<Task>)controller.Index().
            ViewData.Model).ToList();

//            Утверждение

        Assert.AreEqual(result.Count(), 5); Assert.AreEqual("Задача1",
result[0].Name); Assert.AreEqual("Задача2", result[1].Name);
Assert.AreEqual("Задача3", result[2].Name);
    }
    [TestMethod]

    public void Can_Edit_Task()
    {
//        Организация - созданиеимитированногохранилищаданных

        Mock<IRaspRepository>newmock = new Mock<IRaspRepository>();
newmock.Setup(m => m.Task).Returns(new List<Task>
    {

        new Task { TaskId = 1, Name = "Задача1"}, new Task { TaskId =
2, Name = "Задача2"}, new Task { TaskId = 3, Name = "Задача3"},
new Task { TaskId = 4, Name = "Задача4"}, new Task { TaskId = 5,
Name = "Задача5"} });

//        Организация - созданиеконтроллера
        AdminController controller = new
        AdminController(newmock.Object);

//        Действие

        Task task1 = controller.Edit(1).ViewData.Model as Task; Task
task2 = controller.Edit(2).ViewData.Model as Task; Task task3 =
controller.Edit(3).ViewData.Model as Task;

        // Assert

        Assert.AreEqual(1, task1.TaskId);
        Assert.AreEqual(2, task2.TaskId);
        Assert.AreEqual(3, task3.TaskId);
    }

    [TestMethod]
    public void Cannot_Edit_Nonexistent_Task()
    {
//        Организация - созданиеимитированногохранилищаданных

        Mock<IRaspRepository>newmock = new Mock<IRaspRepository>();
newmock.Setup(m => m.Task).Returns(new List<Task>

    {

```

```

        new Task { TaskId = 1, Name = "Задача1"}, new Task { TaskId =
2, Name = "Задача2"}, new Task { TaskId = 3, Name = "Задача3"},
new Task { TaskId = 4, Name = "Задача4"}, new Task { TaskId = 5,
Name = "Задача5"} });

//          Организация - создание контроллера

        AdminController controller = new
AdminController(newmock.Object);

//          Действие
        Task result = controller.Edit(6).ViewData.Model as Task;

//          Assert
    }

    [TestMethod]
    public void Can_Save_Valid_Changes()
    {

//          Организация - создание имитированного хранилища данных
        Mock<IRaspRepository>newmock = new Mock<IRaspRepository>();

//          Организация - создание контроллера

        AdminController controller = new
AdminController(newmock.Object);

//          Организация - создание объекта Task Task task = new Task {
Name = "Test" };

//          Действие - попытка сохранения Задачаа
        ActionResult result = controller.Edit(task);

//          Утверждение - проверка того, что к хранилищу производится
обращение
        newmock.Verify(m => m.SaveTask(task));

//          Утверждение - проверка типа результата метода
        Assert.IsNotInstanceOfType(result, typeof(ViewResult));
    }

    [TestMethod]
    public void Cannot_Save_Invalid_Changes()
    {

//          Организация - создание имитированного хранилища данных
        Mock<IRaspRepository>newmock = new Mock<IRaspRepository>();

//          Организация - создание контроллера

        AdminController controller = new
AdminController(newmock.Object);

```

```

//      Организация - созданиеобъекта Task Task task = new Task {
Name = "Test" };

//      Организация - добавление ошибки в состояние модели
controller.ModelState.AddModelError("error", "error");

//      Действие - попытка сохранения Задачаа

        ActionResult result = controller.Edit(task);

//      Утверждение - проверка того, что обращение к хранилищу НЕ
производится

        newmock.Verify(m => m.SaveTask(It.IsAny<Task>()),
Times.Never());

//      Утверждение - проверка типа результата метода
Assert.IsInstanceOfType(result, typeof(ViewResult));
    }

    [TestMethod]
    public void Can_Delete_Valid_Tasks()
    {

//      Организация - созданиеобъекта Task
Task task = new Task { TaskId = 2, Name = "Задача2" };

//      Организация - созданиеимитированногохранилищаданных

        Mock<IRaspRepository>newmock = new Mock<IRaspRepository>();
newmock.Setup(m => m.Task).Returns(new List<Task>
    {

        new Task { TaskId = 1, Name = "Задача1"}, new Task { TaskId = 2,
Name = "Задача2"}, new Task { TaskId = 3, Name = "Задача3"},

        new Task { TaskId = 4, Name = "Задача4"}, new Task { TaskId =
5, Name = "Задача5" } });

//      Организация - созданиеконтроллера
AdminController controller = new
AdminController(newmock.Object);

//      Действие - удалениеЗадачаа controller.Delete(task.TaskId);

//      Утверждение - проверка того, что метод удаления в
хранилище

//      вызывается для корректного объекта Task newmock.Verify(m
=>m.DeleteTask(task.TaskId));
    }
}
}

```

Листинг Б.5 – Код модуля «Rasp.UnitTests.RaspTests.cs»

```

using System;

using Microsoft.VisualStudio.TestTools.UnitTesting;      using
System.Linq;
using System.Collections.Generic;
using Rasp.Domain.Entities;
using Rasp.Domain.Abstract;

using Moq;
using Rasp.WebUI.Controllers;
using System.Web.Mvc;

using Rasp.WebUI.Models;

namespace Rasp.UnitTests
{
    [TestClass]
    public class RaspTests
    {
        [TestMethod]
        public void Can_Add_New_Lines()
        {
            //      Организация - создание нескольких тестовых Задачаов

            Tasktask1 = new Task { TaskId = 1, Name = "Задача1" }; Task task2
            = new Task { TaskId = 2, Name = "Задача2" };

            //      Организация - создание списка

            Rasp rasp = new Rasp();

            //      Действие rasp.AddItem(task1, 1);

            rasp.AddItem(task2, 1);
            List<RaspLine> results = rasp.Lines.ToList();

            //      Утверждение

            Assert.AreEqual(results.Count(),                2);
            Assert.AreEqual(results[0].Task,                 task1);
            Assert.AreEqual(results[1].Task, task2);
        }

        [TestMethod]

        public void Can_Add_Quantity_For_Existing_Lines()
        {
            //      Организация - создание нескольких тестовых Задачаов

```

```

        Tasktask1 = new Task { TaskId = 1, Name = "Задача1" }; Task task2
= new Task { TaskId = 2, Name = "Задача2" };

//          Организация - создание списка

        Rasp rasp = new Rasp();

//          Действие  rasp.AddItem(task1, 1);  rasp.AddItem(task2, 1);
rasp.AddItem(task1, 5);
        List<RaspLine> results = rasp.Lines.OrderBy(c =>

        c.Task.TaskId).ToList();

// Утверждение
Assert.AreEqual(results.Count(), 2);

Assert.AreEqual(results[0].Quantity, 6);
экземпляровдобавленовкорзину

Assert.AreEqual(results[1].Quantity, 1);

    }

// 6

[TestMethod]
public void Can_Remove_Line()

{
//          Организация - создание нескольких тестовых Задачаов

        Tasktask1 = new Task { TaskId = 1, Name = "Задача1" }; Task task2
= new Task { TaskId = 2, Name = "Задача2" }; Task task3 = new Task {
TaskId = 3, Name = "Задача3" };

//          Организация - создание списка

        Rasp rasp = new Rasp();

//          Организация - добавление нескольких Задачаов в корзину
rasp.AddItem(task1, 1);

        rasp.AddItem(task2, 4);          rasp.AddItem(task3, 2);
rasp.AddItem(task2, 1);

//          Действие
        rasp.RemoveLine(task2);

//          Утверждение
Assert.AreEqual(rasp.Lines.Where(c => c.Task ==

        task2).Count(), 0);
Assert.AreEqual(rasp.Lines.Count(), 2);

```

```

    }

    [TestMethod]
    public void Calculate_Rasp_Total()
    {
//        Организация - создание нескольких тестовых Задачаов

        Tasktask1 = newTask { TaskId = 1, Name = "Задача1", Price
        100 };

        Task task2 = new Task { TaskId = 2, Name = "Задача2", Price =
55 };

//        Организация - создание списка

        Rasp rasp = new Rasp();

//        Действие
        rasp.AddItem(task1, 1);

        rasp.AddItem(task2, 1);
        rasp.AddItem(task1, 5);
        decimal result = rasp.ComputeTotalValue();

//        Утверждение
        Assert.AreEqual(result, 655);
    }

    [TestMethod]
    public void Can_Clear_Contents()
    {
//        Организация - создание нескольких тестовых Задачаов

        Tasktask1 = newTask { TaskId = 1, Name = "Задача1", Price
        100 };

        Task task2 = new Task { TaskId = 2, Name = "Задача2", Price =
55 };

//        Организация - создание списка

        Rasp rasp = new Rasp();

//        Действие

        rasp.AddItem(task1, 1);
        rasp.AddItem(task2, 1);
        rasp.AddItem(task1, 5);
        rasp.Clear();

//        Утверждение
        Assert.AreEqual(rasp.Lines.Count(), 0);
    }

```

```

    }

    [TestMethod]

    public void Can_Add_To_Rasp()
    {
        //      Организация - созданиеимитированногохранилища

        Mock<IRaspRepository>newmock = new Mock<IRaspRepository>();
        newmock.Setup(m => m.Task).Returns(new List<Task> {

            new Task {TaskId = 1, Name = "Задача1", Category = "Kat1"},
        }.AsQueryable());

        //      Организация - созданиесписка
        Rasprasp = newRasp();

        //      Организация - созданиеконтроллера
        RaspController controller = new
            RaspController(newmock.Object, null);

        //      Действие - добавить Задача в корзину
        controller.AddToRasp(rasp, 1, null);

        //      Утверждение

        Assert.AreEqual(rasp.Lines.Count(), 1);
        Assert.AreEqual(rasp.Lines.ToList()[0].Task.TaskId, 1);

    }

    ///      <summary>

    ///      После добавления Задачаа в корзину, должно быть
    ///      перенаправление на страницу списка
    ///      </summary>
    [TestMethod]

    public void Adding_Task_To_Rasp_Goes_To_Rasp_Screen()
    {
        //      Организация - созданиеимитированногохранилища

        Mock<IRaspRepository>newmock = new Mock<IRaspRepository>();
        newmock.Setup(m => m.Task).Returns(new List<Task> {

            new Task {TaskId = 1, Name = "Задача1", Category = "Kat1"},
        }.AsQueryable());

        //      Организация - созданиесписка
        Rasprasp = newRasp();

        //      Организация - созданиеконтроллера
        RaspController controller = new

```

```

        RaspController(newmock.Object, null);

//        Действие - добавитьЗадачавкорзину

        RedirectToRouteResult result = controller.AddToRasp(rasp, 2,
"myUrl");

//        Утверждение

        Assert.AreEqual(result.RouteValues["action"], "Index");
Assert.AreEqual(result.RouteValues["returnUrl"], "myUrl");
    }

//        Проверяем URL [TestMethod]
public void Can_View_Rasp_Contents()

{
//        Организация - создание списка
Rasp rasp = new Rasp();

//        Организация - созданиеконтроллера
RaspController target = new RaspController(null, null);

//        Действие - вызов метода действия Index()
RaspIndexViewModel result
= (RaspIndexViewModel)target.Index(rasp,
"myUrl").ViewData.Model;

//        Утверждение

        Assert.AreSame(result.Rasp, rasp);
Assert.AreEqual(result.ReturnUrl, "myUrl");
    }

    [TestMethod]

    public void Cannot_Checkout_Empty_Rasp()
    {
//        Организация - созданиеимитированногообработчиказаказов

        Mock<IOrderProcessor>newmock = new Mock<IOrderProcessor>();

//        Организация - создание пустой списка
Rasp rasp = new Rasp();

//        Организация - созданиедеталейодоставке
ShippingDetailsshippingDetails = newShippingDetails();

//        Организация - созданиеконтроллера

        RaspController controller = new RaspController(null,
newmock.Object);

//        Действие

```

```

        ActionResult result = controller.Checkout(rasp,
shippingDetails);

//      Утверждение - проверка, что заказ не был передан
обработчику

        newmock.Verify(m => m.ProcessOrder(It.IsAny<Rasp>(),
It.IsAny<ShippingDetails>()),
        Times.Never());

//      Утверждение - проверка, что метод вернул стандартное
представление
        Assert.AreEqual("", result.ViewName);

//      Утверждение - проверка, что-представлению передана
неверная модель

        Assert.AreEqual(false, result.ViewData.ModelState.IsValid); }

[TestMethod]
public void Can_Checkout_And_Submit_Order()
{
//      Организация - созданиеимитированногообработчиказаказов

        Mock<IOrderProcessor>newmock = new Mock<IOrderProcessor>();

//      Организация - создание списка с элементом

        Rasp rasp = new Rasp();
        rasp.AddItem(new Task(), 1);

//      Организация - создание контроллера

        RaspController controller = new RaspController(null,
newmock.Object);

//      Действие - попыткапереходакоплате

        ActionResult result = controller.Checkout(rasp, new
ShippingDetails());

//      Утверждение - проверка, что заказ передан обработчику
newmock.Verify(m => m.ProcessOrder(It.IsAny<Rasp>(),
        It.IsAny<ShippingDetails>()),

        Times.Once());

//      Утверждение - проверка, что метод возвращает
представление

        Assert.AreEqual("Completed", result.ViewName);

```

```
//      Утверждение - проверка, что представлению передается
допустимая модель
    Assert.AreEqual(true, result.ViewData.ModelState.IsValid);
}
}

}
```

Листинг Б.6 – Код модуля «Rasp.UnitTests.ImageTests.cs»

```
using Microsoft.VisualStudio.TestTools.UnitTesting; using Moq;

using Rasp.Domain.Abstract;
using Rasp.Domain.Entities;
using Rasp.WebUI.Controllers;

using System.Linq;
using System.Web.Mvc;
using System.Collections.Generic;

namespace Rasp.UnitTests
{
    [TestClass]

    public class ImageTests

    {
        [TestMethod]
        public void Can_Retrieve_Image_Data()

        {

//      Организация - создание объекта Task с данными изображения
        Task task = new Task

        {

            TaskId = 2, Name = "Задача2",

            ImageData = new byte[] { }, ImageMimeType = "image/png" };

//      Организация - создание имитированного хранилища

        Mock<IRaspRepository>newmock = new Mock<IRaspRepository>();
        newmock.Setup(m => m.Task).Returns(new List<Task> {

            new Task {TaskId = 1, Name = "Задача1"}, task,

            new Task {TaskId = 3, Name = "Задача3"} }.AsQueryable());

//      Организация - создание контроллера
```

```

        RaspController          controller          =          new
RaspController(newmock.Object);

//          Действие - вызов метода действия GetImage()
ActionResult result = controller.GetImage(2);

//          Утверждение
Assert.IsNotNull(result);

Assert.IsInstanceOfType(result, typeof(FileResult));
Assert.AreEqual(task.ImageMimeType,
((FileResult)result).ContentType);
}

[TestMethod]
public void Cannot_Retrieve_Image_Data_For_Invalid_ID()
{
//          Организация - созданиеимитированногохранилища

Mock<IRaspRepository>newmock = new Mock<IRaspRepository>();
newmock.Setup(m => m.Task).Returns(new List<Task> {

    new Task {TaskId = 1, Name = "Задача1"}, new Task {TaskId =
2, Name = "Задача2"} }.AsQueryable());

//          Организация - созданиеконтроллера

        RaspController          controller          =          new
RaspController(newmock.Object);

//          Действие - вызов метода действия GetImage()
ActionResult result = controller.GetImage(10);

//          Утверждение
Assert.IsNull(result);

}
}
}

```

Листинг Б.7 –Код модуля «Rasp.UnitTests.UnitTest1.cs»

```

using System;
using System.Collections.Generic;
using System.Linq;

using System.Web.Mvc;

using Microsoft.VisualStudio.TestTools.UnitTesting; using Moq;
using Rasp.Domain.Abstract;

```

```

using Rasp.Domain.Entities;
using Rasp.WebUI.Controllers;
using Rasp.WebUI.Models;

using Rasp.WebUI.HtmlHelpers;

namespace Rasp.UnitTests
{
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void Can_Paginate()
        {
            //      Организация (arrange)

            Mock<IRaspRepository>newmock = new Mock<IRaspRepository>();
            newmock.Setup(m => m.Task).Returns(new List<Task> {

                new Task { TaskId = 1, Name = "Задача1"}, new Task { TaskId =
2, Name = "Задача2"}, new Task { TaskId = 3, Name = "Задача3"},
new Task { TaskId = 4, Name = "Задача4"}, new Task { TaskId = 5,
Name = "Задача5"} });

            RaspController controller = new
RaspController(newmock.Object);

            controller.pageSize = 3;

            //      Действие (act) TasksListViewModel result =
            (TasksListViewModel)controller.List(null, 2).Model;

            //      Утверждение

            List<Task> tasks = result.Tasks.ToList();
            Assert.IsTrue(tasks.Count == 2); Assert.AreEqual(tasks[0].Name,
"Задача4"); Assert.AreEqual(tasks[1].Name, "Задача5");

        }
        [TestMethod]
        public void Can_Generate_Page_Links()
        {

            //      Организация - определение вспомогательного метода HTML -
            это необходимо

            //      для применения расширяющего метода
            HtmlHelper myHelper = null;

            //      Организация - создание объекта PagingInfo
            PagingInfo pagingInfo = new PagingInfo
            {

```

```

        CurrentPage = 2, TotalItems = 28, ItemsPerPage = 10 };

//      Организация - настройка делегата с помощью лямбда-
выражения
        Func<int, string> pageUrlDelegate = i => "Page" + i;

//      Действие

        MvcPhpString      result      =      myHelper.PageLinks(pagingInfo,
pageUrlDelegate);

//      Утверждение
        Assert.AreEqual(@"<a class=""btn btn-default""
href=""Page1"">1</a>"
        @"<a      class=""btn      btn-default      btn-primary      selected""
href=""Page2"">2</a>"
        @"<a      class=""btn      btn-default""      href=""Page3"">3</a>",
result.ToString());
    }
    [TestMethod]

    public void Can_Send_Pagination_View_Model()
    {
        // Организация (arrange)

        Mock<IRaspRepository>newmock = new Mock<IRaspRepository>();
        newmock.Setup(m => m.Task).Returns(new List<Task>
        {
            new Task { TaskId = 1, Name = "Задача1"}, new Task { TaskId =
2, Name = "Задача2"}, new Task { TaskId = 3, Name = "Задача3"},
new Task { TaskId = 4, Name = "Задача4"}, new Task { TaskId = 5,
Name = "Задача5" } });
        RaspController controller = new
RaspController(newmock.Object);

        controller.pageSize = 3;

        // Act

        TasksListViewModel result
        = (TasksListViewModel)controller.List(null, 2).Model;

        // Assert

        PagingInfo      pageInfo      =      result.PagingInfo;
        Assert.AreEqual(pageInfo.CurrentPage,                2);
        Assert.AreEqual(pageInfo.ItemsPerPage,                3);
        Assert.AreEqual(pageInfo.TotalItems,                  5);
        Assert.AreEqual(pageInfo.TotalPages, 2); }

```

```

[TestMethod]
public void Can_Create_Categories()

{
//      Организация - созданиеимитированногохранилища

    Mock<IRaspRepository>newmock = new Mock<IRaspRepository>();
newmock.Setup(m => m.Task).Returns(new List<Task> {
    new Task { TaskId = 1, Name = "Задача1",
    Category="Мат.платы"},

    new Task { TaskId = 2, Name = "Задача2", Category="Задачи"},

    new Task { TaskId = 3, Name = "Задача3", Category=" Задача
"}, new Task { TaskId = 4, Name = "Задача4", Category=" Задача "},
});

//      Организация - созданиеконтроллера
NavController target = new NavController(newmock.Object);

//      Действие - получениенаборакатегорий
List<string> results =

    ((IEnumerable<string>)target.Menu().Model).ToList();

//      Утверждение

    Assert.AreEqual(results.Count(), 3); Assert.AreEqual(results[1],
"Задачи"); Assert.AreEqual(results[0], " Задачи ");
Assert.AreEqual(results[2], " Задачи ");
}

[TestMethod]
public void Indicates_Selected_Category()

{
//      Организация - созданиеимитированногохранилища

    Mock<IRaspRepository>newmock = new Mock<IRaspRepository>();
newmock.Setup(m => m.Task).Returns(new Task[] {

    new Task { TaskId = 1, Name = "Задача1",
    Category=" Задачи "},

    new Task { TaskId = 2, Name = "Задача2", Category=" Задачи "
});

// Организация - созданиеконтроллера

NavController target = new NavController(newmock.Object);

//      Организация - определение выбранной категории string
categoryToSelect = "Задачи";

```

```

//      Действие
string result =

target.Menu(categoryToSelect).ViewBag.SelectedCategory;

//      Утверждение

Assert.AreEqual(categoryToSelect, result);
}

[TestMethod]

public void Generate_Category_Specific_Task_Count()
{
///      Организация (arrange)

Mock<IRaspRepository>newmock = new Mock<IRaspRepository>();
newmock.Setup(m => m.Task).Returns(new List<Task> {

    new Task { TaskId = 1, Name = "Задача1", Category="Cat2"},
    new Task { TaskId = 2, Name = "Задача2", Category="Cat3"}, new
Task { TaskId = 3, Name = "Задача3", Category="Cat2"}, new Task {
TaskId = 4, Name = "Задача4", Category="Cat1"}, new Task { TaskId
= 5, Name = "Задача5", Category="Cat1"} });

    RaspController controller = new RaspController(mock.Object);
    controller.pageSize = 3;

//      Действие - тестирование счетчиков Задачаов для различных
категорий

        int                                res1                                =
((TasksListViewModel)controller.List("Cat1").Model).PagingInfo.
TotalItems;

        int                                res2                                =
((TasksListViewModel)controller.List("Cat2").Model).PagingInfo.
TotalItems;

        int                                res3                                =
((TasksListViewModel)controller.List("Cat3").Model).PagingInfo.
TotalItems;

        int                                resAll                             =
((TasksListViewModel)controller.List(null).Model).PagingInfo.To
talItems;

//      Утверждение
Assert.AreEqual(res1, 2);

Assert.AreEqual(res2, 2);
Assert.AreEqual(res3, 1);
Assert.AreEqual(resAll, 5);}

```

Приложение В
(рекомендуемое)
Результаты тестирования

