СОДЕРЖАНИЕ

[Введение 4](#_Toc145008152)

[1 ХАРАКТЕРИСТИКИ ПРЕДПРИЯТИЯ 5](#_Toc145008153)

[1.1 История развития, общие сведения о предприятии 5](#_Toc145008154)

[1.2 Организационная структура предприятия 5](#_Toc145008155)

[2 АНАЛИЗ функционирования структурных подразделений 7](#_Toc145008156)

[2.1 Анализ функций предприятия 7](#_Toc145008157)

[2.2 Анализ функций и задач IT-отдела 7](#_Toc145008158)

[3 НОРМАТИВНАЯ ДОКУМЕНТАЦИЯ 9](#_Toc145008159)

[3.1 Содержание должностных инструкций 9](#_Toc145008160)

[3.2 Модели жизненного цикла программного обеспечения 11](#_Toc145008161)

[4 ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, ИСПОЛЬЗУЕМОЕ в ОРГАНИЗАЦИИ 13](#_Toc145008162)

[4.1 Среды разработки и языки программирования, используемые в организации 13](#_Toc145008163)

[4.2 Операционные системы, используемые в организации 13](#_Toc145008164)

[4.3 Требования, предъявляемые к разрабатываемому программному продукту 13](#_Toc145008165)

[5 ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ 14](#_Toc145008166)

[5.1 Постановка задачи 14](#_Toc145008167)

[5.2 Выбор и обоснование выбора языка программирования 14](#_Toc145008168)

[5.3 Выбор средств разработки программы 14](#_Toc145008169)

[5.4 Проектирование функциональной структуры 14](#_Toc145008170)

[5.5 Реализация и тестирование программы 15](#_Toc145008171)

[Заключение 21](#_Toc145008172)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 22](#_Toc145008173)

[ПРИЛОЖЕНИЕ А(обязательное) 23](#_Toc145008174)

# Введение

Данный отчёт представляет собой результат выполнения производственной практики в компании «Andersen». Основной целью этой практики было приобретение профессиональных навыков, соответствующих специализации, а также получение опыта работы внутри компании. Помимо этого, практика также предоставила возможность закрепить, расширить и систематизировать знания, полученные в ходе обучения по специальным дисциплинам.

В процессе практики необходимо было ознакомиться с организационной структурой предприятия, изучить функциональные обязанности различных подразделений, оценить степень автоматизации бизнес-процессов компании, а также ознакомиться с предоставляемыми компанией услугами.

Кроме того, одной из важных задач практики было разработать специализированное программное приложение, которое было создано с учетом особенностей предметной области и результатов анализа теоретического материала.

В рамках практики было спроектировано и разработано интерактивное веб-приложение, предназначенное для управления списком дел. Это приложение разработано с учетом современных требований и потребностей, связанных с управлением задачами и делами.

Таким образом, прохождение производственной практики в компании «Andersen» позволило не только приобрести ценный опыт, но и внести вклад в развитие программного обеспечения, удовлетворяющего потребностям современного бизнеса и повседневной жизни.

# ХАРАКТЕРИСТИКИ ПРЕДПРИЯТИЯ

## История развития, общие сведения о предприятии

Andersen - это одна из ведущих мировых айти-компаний, специализирующихся на разработке программного обеспечения и консалтинге в сфере информационных технологий. Она имеет богатую историю развития, начиная с её создания и становления.

Andersen была основана в 2007 году. Основателями компании стали Антон Янчук и Владимир Большаков. Сначала компания была маленькой командой разработчиков, работающей над веб-проектами.

За первые несколько лет существования, Andersen активно росла и расширялась. Она начала предоставлять услуги разработки программного обеспечения и консультаций для клиентов. Компания быстро набирала клиентскую базу благодаря своей команде высококвалифицированных разработчиков и инженеров.

С течением времени, Andersen расширила свой спектр услуг, добавив консалтинг по цифровой трансформации, разработку и поддержку мобильных приложений, искусственный интеллект, аналитику данных и многое другое. Это позволило компании более глубоко взаимодействовать с клиентами и предоставлять комплексные решения.

В последние годы Andersen активно участвует в развитии айти-сообщества и инновационных проектов. Компания вступила в стратегические партнерства с ведущими компаниями технологической отрасли и активно поддерживает стартапы и мероприятия, связанные с технологиями.

## Организационная структура предприятия

OOO «Andersen» имеет плоскую организационную структуру организации.

Плоская организация (также известная как горизонтальная организация или плоская иерархия) имеет организационную структуру с небольшим или отсутствующим уровнем среднего звена между персоналом и руководителями. Структура организации относится к характеру распределения подразделений и должностей внутри нее, а также к характеру взаимоотношений между этими подразделениями и должностями. Высокие и беспристрастные организации различаются в зависимости от того, сколько уровней управления присутствует в организации, и на сколько наделены управляющие. Преобразование высоко иерархической организации в плоскую организацию называется задержкой.

В плоских организациях количество людей, непосредственно контролируемых каждым менеджером, велико, а количество людей в цепочке командования выше одного - мало. Менеджер в плоской организации несет большую ответственность, чем менеджер в высокой организации, потому что непосредственно под ним находится большее количество людей, которые зависят от направления, помощи и поддержки. Более того, менеджеры в плоской организации меньше полагаются на руководство со стороны начальства, потому что количество вышестоящих над менеджером ограничено.

Эмпирические данные Гизели и Джонсона позволяют предположить, что количество независимых менеджеров в плоских организациях, полученных в результате плоской организационной структуры, удовлетворяет многие их потребности с точки зрения автономии и самореализации. Идея, лежащая в основе неорганизованных организаций, заключается в том, что хорошо обученные работники будут более продуктивными, когда будут более непосредственно вовлечены в процесс принятия решений, а не будут тщательно контролироваться многими уровнями управления. Такая структура обычно возможна только в небольших организациях или отдельных подразделениях в более крупных организациях. Достигнув критического размера, организации могут сохранить упорядоченную структуру, но не могут поддерживать абсолютно ровные взаимоотношения между менеджером и персоналом, не влияя на производительность. Определенные финансовые обязанности могут также потребовать более традиционной структуры. Компания не должна была бы давать повышение или продвижение по службе, основываясь на длине обслуживания, но на более высокой производительности. Кроме того, исключение некоторых отделов из фонда оплаты труда означает экономию денег. Некоторые компании теоретизируют, что плоские организации становятся более традиционно иерархичными, когда они начинают ориентироваться на производительность.

Модель плоской организации способствует вовлечению сотрудников посредством децентрализованного процесса принятия решений. Повышая уровень ответственности базовых сотрудников и устраняя уровни среднего звена, комментарии и отзывы быстрее доходят до всего персонала, участвующего в принятии решений. Ожидаемый ответ на отзывы клиентов становится более быстрым.[1]

# АНАЛИЗ функционирования структурных подразделений

## 2.1 Анализ функций предприятия

ООО «Andersen» основывает свою деятельность на богатом опыте и глубоких знаниях, создавая инновационные технологии и предоставляя клиентам решения для глобальных бизнес-задач. Достижения и экспертиза охватывают различные области, включая:

– внедрение ERP, PLM, CRM, SCM решений и систем аналитики, стратегического планирования и бюджетирования в ряде отраслей;

– анализ инфраструктуры и информационных ресурсов, проектирование и реинжиниринг бизнес-процессов, управление проектами модернизации и развития информационных систем

– разработка по заказам производителей программного обеспечения для систем корпоративного планирования (ERP), управления жизненным циклом изделий (PLM), корпоративных информационных порталов (EIP), систем управления отношениями с клиентами (CRM);

– разработка приложений, соответствующих требованиям новейших сервис-ориентированных архитектур (SOA - service oriented architecture);

– создание и развертывание электронных систем управления закупками и сбытом;

– построение порталов крупных предприятий и холдингов с развитыми средствами анализа данных и управления знаниями;

– интеграция приложений в распределенных системах (в том числе насчитывающих сотни производственных площадок, сотни унаследованных приложений и десятки ERP-систем), проектирование, консолидация и настройка корпоративных справочников и каталогов;

Высококвалифицированные сотрудники и эффективные процессы разработки позволяют предоставлять клиентам наилучшие IT-решения, объединяя в себе преимущества индивидуальных заказных проектов и масштабируемых продуктов.

## 2.2 Анализ функций и задач IT-отдела

Компания «Andersen» специализируется на разработке программного обеспечения и выделяется среди конкурентов благодаря своим уникальным характеристикам.

Она обладает превосходно отлаженными рабочими процессами, которые гарантируют высокую эффективность в выполнении проектов. Команда состоит из высококвалифицированных специалистов, работающих в штате компании, что обеспечивает профессионализм и надежность.

За годы работы успешно завершили множество проектов разной сложности и в разных отраслях. Располагают ноу-хау и наработками, которые позволяют нам создавать инновационные решения.

У них есть собственный отдел тестирования, что помогает гарантировать высокое качество разрабатываемого программного обеспечения и снижать риски связанные с его работой в нештатных ситуациях.

Компания работает с разнообразными клиентами, включая как малые и крупные частные предприятия, так и государственные организации. Их богатый опыт накоплен в разных областях, таких как ритейл, медицина, мультимедиа и государственное управление.

Одним из основных направлений деятельности является разработка заказного программного обеспечения. Они берут на себя полный контроль над процессом, начиная с формулирования требований и анализа потребностей клиента, и заканчивая созданием визуальных моделей и предоставлением экспертных рекомендаций.

# 3 НОРМАТИВНАЯ ДОКУМЕНТАЦИЯ

## 3.1 Содержание должностных инструкций

Должностная инструкция инженера-программиста. Общие положения:

1. Вакансия ведущего инженера-программиста оформляется приказом руководителя при приеме на работу и также уведомлением при увольнении.
2. Для занятия должности ведущего инженера-программиста требуется наличие высшего образования в соответствующей области квалификации.
3. В своей деятельности инженер-программист руководствуется:

* нормативными документами по вопросам выполняемой работы;
* уставом организации;
* правилами трудового распорядка;
* приказами и распоряжениями руководителя организации (непосредственного руководителя).

1. Ведущий инженер-программист должен знать:

* руководящие и нормативные материалы, регламентирующие методы разработки алгоритмов, программ и использования вычислительной техники при обработке информации;
* основные принципы структурного и объектно-ориентированного программирования;
* виды программного обеспечения;
* стандарты программной документации;
* основные методы, средства и методологии разработки программного обеспечения;
* стандартные алгоритмы;
* языки программирования;
* технологию автоматизированной обработки информации;
* виды и порядок оформления технической документации;
* стандарты качества программного обеспечения;
* основные общепринятые технологии в области разработки программных продуктов;
* технико-эксплуатационные характеристики, конструктивные особенности, назначение и режимы работы оборудования, правила его технической эксплуатации;
* основы организации труда и управления;
* основы законодательства о труде;
* правила и нормы охраны труда и пожарной безопасности.

1. Во время отсутствия ведущего инженера-программиста его обязанности выполняет в установленном порядке назначаемый заместитель, несущий полную ответственность за надлежащее исполнение возложенных на него обязанностей.

Должностные обязанности. Для выполнения возложенных на него функций ведущий инженер-программист обязан:

* На основе анализа математических моделей и алгоритмов решения научных, прикладных, экономических и других задач разрабатывать программы, обеспечивающие возможность выполнения средствами вычислительной техники алгоритма и поставленной задачи.
* Участвовать в выборе языка программирования для описания алгоритмов и структур данных.
* Разрабатывать технологию решения задачи на всех этапах.
* Определять информацию, подлежащую обработке средствами вычислительной техники, ее объемы, структуру, макеты и схемы ввода, обработки, хранения и вывода, методы ее контроля.
* Определять объем и содержание данных контрольных примеров, обеспечивающих наиболее полную проверку соответствия программ их функциональному назначению.
* Осуществлять запуск отлаженных программ и ввод исходных данных, определяемых условиями поставленных задач.
* Определять возможность использования готовых программных продуктов.
* Разрабатывать, отлаживать, анализировать и оптимизировать программный код на основе готовых спецификаций.
* Интегрировать программные компоненты.
* Проводить проверку программ на основе логического анализа.
* Проводить отладку разработанных программ, корректировать их в процессе стабилизации и сопровождения.
* Выполнять работу по унификации процессов разработки программы.
* Разрабатывать инструкции по работе с программами, оформлять необходимую техническую документацию.
* Разрабатывать и внедрять методы автоматизации программирования.
* Участвовать в сопровождении программного обеспечения.
* Оказывать помощь специалистам на различных стадиях разработки программного обеспечения при сборе и документировании требований пользователя, в разработке спецификации.
* Вести и представлять установленную отчетность.
* Своевременно и точно исполнять производственные приказы, задания, указания и распоряжения руководства.
* Соблюдать установленные на предприятии Правила внутреннего трудового распорядка.
* Оказывать содействие и сотрудничать с нанимателем в деле обеспечения здоровых и безопасных условий труда, немедленно сообщать непосредственному руководителю о каждом случае производственного травматизма и профессионального заболевания, а также о чрезвычайных ситуациях, которые создают угрозу здоровью и жизни для него и окружающих, обнаруженных недостатках и нарушениях охраны.[2]
* Принимать необходимые меры по ограничению развития аварийной ситуации и ее ликвидации.

Права. Ведущий инженер-программист имеет право:

* Знакомиться с проектами решений руководства организации, касающимися его деятельности.
* Вносить на рассмотрение руководства по совершенствованию информационной системы организации в целом.
* Получать от руководителей структурных подразделений, специалистов информацию и документы, необходимые для выполнения своих должностных обязанностей.
* Привлекать специалистов организации для решения возложенных на него обязанностей (если это предусмотрено положениями о структурных подразделениях, если нет – с разрешения руководителя организации).
* Требовать от руководства организации оказания содействия в исполнении своих должностных обязанностей и прав.
* Принимать участие в обсуждении вопросов охраны труда, выносимых на рассмотрение собраний (конференций) трудового коллектива (профсоюзной организации)

Оценка работы и ответственность. Работу ведущего инженера-программиста оценивает непосредственный руководитель (иное должностное лицо) Ведущий инженер-программист несёт ответственность:

* За неисполнение (ненадлежащее поведение) своих должностных обязанностей, предусмотренных настоящей должностной инструкцией, – в пределах, определенных действующим трудовым законодательством РБ.
* За совершенные в процессе осуществления своей деятельности правонарушения – в пределах, определенных действующим административным, уголовным и гражданским законодательством РБ.
* За причинение материального ущерба – в пределах, определенных действующим трудовым, уголовным и гражданским законодательством РБ.
* За несоблюдение правил и норм охраны труда, техники безопасности, производственной санитарии и противопожарной защиты – в соответствии с требованиями нормативных правовых актов РБ и локальных актов.

## 3.2 Модели жизненного цикла программного обеспечения

Жизненный цикл программного обеспечения (ПО) представляет собой последовательность событий, которые происходят при создании и использовании программных продуктов. Он может быть представлен в виде различных моделей. Эти модели включают инженерный подход, учет спецификации задачи и современные технологии быстрой разработки.

В прежние времена, программные приложения были едиными и для их разработки применялся каскадный метод. Этот метод предполагал разделение разработки на этапы, с переходом к следующему этапу только после завершения текущего. Каждый этап завершался созданием полного комплекта документации.

Стандарт ISO 12207 является международным стандартом, который полно описывает процессы жизненного цикла программного обеспечения, технологию разработки и обеспечения качества сложных программных средств. Жизненный цикл программного обеспечения включает этапы от подготовки технического задания до завершения эксплуатации.

Процессы разработки программного обеспечения начинаются с инициации проекта, анализа концепции и требований, и включают создание плана проекта, выбор средств разработки и документирование всех этапов.

Процессы сопровождения программного обеспечения включают анализ сообщений об ошибках, предложений по модификации программы, их оценку и тестирование изменений.[3]

Процессы документирования включают планирование и создание документации, а также обеспечение качества программного обеспечения.

Верификация и аттестация обеспечивают соответствие программного продукта требованиям и гарантируют его безопасное и надежное использование.

Таким образом, жизненный цикл программного обеспечения включает в себя множество процессов, начиная с разработки и заканчивая сопровождением и обеспечением качества, что позволяет создавать и поддерживать сложные программные продукты.

# 4 ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, ИСПОЛЬЗУЕМОЕ 1в ОРГАНИЗАЦИИ

## 4.1 Среды разработки и языки программирования, используемые в организации

Для разработки программного обеспечения в отделе используются такие среды разработки как: Microsoft Visual Studio, 1C-Bitrix, IntelliJ IDEA, VS Code, PhpStorm.

Средства и технологии, которые используются для разработки ПО:

1. Языки программирования (C, C++, C#, PHP, Java, JavaScript, Python, Ruby, Scala, Golang)
2. Языки разметки (HTML, XML)
3. Фреймворки (Angular, React, Vue, ASP.NET Core, Unity, Xamarin, iOS, Android)
4. СУБД (MS SQL, MySQL, LINQ to SQL)

## Операционные системы, используемые в организации

Компания Andersen использует операционные системы семейства Windows, Mac OS и Linux, на которые оформлены соответствующие лицензии.

## 4.3 Требования, предъявляемые к разрабатываемому программному продукту

К главным требованиям, предъявляемым к разрабатываемому продукту, относятся:

* Правильность – функционирование в соответствии с техническим заданием.
* Универсальность – обеспечение правильной работы при любых допустимых данных и защиты от неправильных данных.
* Надежность (помехозащищенность) – обеспечение полной повторяемости результатов, т. е. обеспечение их правильности при наличии различного рода сбоев.
* Защищенность – обеспечение конфиденциальности информации.
* Аппаратная совместимость – возможность совместного функционирования с некоторым оборудованием.

# 5 ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

## 5.1 Постановка задачи

За время прохождения практики необходимо было разработать веб-приложение, а именно предоставить сайт для составления списка дел.

## 5.2 Выбор и обоснование выбора языка программирования

Согласно требованиям к программному средству, данная разрабатываемая программа должна представлять собой веб-сайт.

Для создания веб-сайта был использован язык JavaScript.

JavaScript — это высокоуровневый язык программирования, который часто используется для создания интерактивных веб-сайтов и веб-приложений. VS Code обеспечивает надежную поддержку для разработки на JavaScript, и в нем можно легко создавать, отлаживать и управлять проектами, написанными на этом языке.

## 5.3 Выбор средств разработки программы

В качестве интегрированной среды разработки была выбрана Visual Studio Code.

Visual Studio Code (VS Code) — это интегрированная среда разработки (IDE), разработанная Microsoft, которая предоставляет множество инструментов и функций для разработки различных языков программирования, включая JavaScript.

## 5.4 Проектирование функциональной структуры

Проектирование функциональной структуры интерактивного приложения включает в себя определение функций и их взаимодействие для обеспечения работы приложения. Вот функциональная структура для такого приложения:

– добавление задачи;

– отметка задачи как выполненной;

– удаление задачи;

– фильтрация задач;

– сортировка задач;

– загрузка задач из локального хранилища;

– интерфейс;

– обработка событий.

## 5.5 Реализация и тестирование программы

Для реализации интерактивного приложения потребовались HTML, CSS и JavaScript:

Листинг 5.1 – Реализация веб-сайта

<!DOCTYPE html>

<html>

<head>

<title>Список дел</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

background-color: #f4f4f4;

}

h1 {

text-align: center;

color: #333;

}

#taskInput {

width: 70%;

padding: 8px;

margin-right: 10px;

font-size: 16px;

border: 1px solid #ccc;

border-radius: 4px;

}

button {

padding: 8px 16px;

font-size: 16px;

border: none;

background-color: #007bff;

color: #fff;

border-radius: 4px;

cursor: pointer;

}

button:hover {

background-color: #0056b3;

Продолжение Листинга 5.1

}

ul {

list-style-type: none;

padding: 0;

}

li {

background-color: #fff;

margin: 5px 0;

padding: 10px;

border: 1px solid #ddd;

border-radius: 4px;

display: flex;

justify-content: space-between;

align-items: center;

}

select {

font-size: 16px;

padding: 5px;

}

label {

font-weight: bold;

}

</style>

</head>

<body>

<h1>Список дел</h1>

<input type="text" id="taskInput" placeholder="Добавить новую задачу">

<button onclick="addTask()">Добавить</button>

<div>

<label for="filter">Фильтр:</label>

<select id="filter" onchange="filterTasks()">

<option value="all">Все</option>

<option value="active">Активные</option>

<option value="completed">Выполненные</option>

</select>

</div>

<div>

<label for="sort">Сортировка:</label>

<select id="sort" onchange="sortTasks()">

<option value="alphabetical">По алфавиту</option>

<option value="date">По дате</option>

<option value="status">По статусу</option>

</select>

</div>

<ul id="taskList">

<!-- Здесь будут отображаться задачи -->

</ul>

<script>

function addTask() {

Продолжение Листинга 5.1

var taskInput = document.getElementById('taskInput');

var taskText = taskInput.value.trim();

if (taskText === '') {

alert('Введите задачу!');

return;

}

var taskList = document.getElementById('taskList');

var newTaskItem = document.createElement('li');

newTaskItem.innerHTML = `

${taskText}

<button onclick="completeTask(this)">Завершить</button>

<button onclick="removeTask(this)">Удалить</button>`;

taskList.appendChild(newTaskItem);

taskInput.value = '';

saveTasks();

}

function completeTask(button) {

var taskItem = button.parentElement;

taskItem.style.textDecoration = 'line-through';

button.disabled = true;

saveTasks();

}

function removeTask(button) {

var taskItem = button.parentElement;

taskItem.remove();

saveTasks();

}

function filterTasks() {

var filter = document.getElementById('filter').value;

var taskList = document.getElementById('taskList');

var tasks = taskList.children;

for (var i = 0; i < tasks.length; i++) {

var taskItem = tasks[i];

var isCompleted = taskItem.style.textDecoration === 'line-through';

if ((filter === 'active' && isCompleted) || (filter === 'completed' && !isCompleted) || (filter === 'all')) {

taskItem.style.display = 'none';

} else {

taskItem.style.display = 'list-item';

}

}

}

function sortTasks() {

var sort = document.getElementById('sort').value;

var taskList = document.getElementById('taskList');

var tasks = Array.from(taskList.children);

tasks.sort(function (a, b) {

Продолжение Листинга 5.1

switch (sort) {

case 'alphabetical':

return a.textContent.localeCompare(b.textContent);

case 'date':

return a.dataset.timestamp - b.dataset.timestamp;

case 'status':

return (a.style.textDecoration === 'line-through' ? 1 : -1);

default:

return 0;

}

});

taskList.innerHTML = '';

tasks.forEach(function (task) {

taskList.appendChild(task);

});

}

window.onload = function () {

var savedTasks = JSON.parse(localStorage.getItem('tasks')) || [];

var taskList = document.getElementById('taskList');

savedTasks.forEach(function (taskText) {

var newTaskItem = document.createElement('li');

newTaskItem.innerHTML = `

${taskText}

<button onclick="completeTask(this)">Завершить</button>

<button onclick="removeTask(this)">Удалить</button>

`;

taskList.appendChild(newTaskItem);

});

}

function saveTasks() {

var taskList = document.getElementById('taskList');

var tasks = [];

for (var i = 0; i < taskList.children.length; i++) {

var taskText = taskList.children[i].textContent.trim();

tasks.push(taskText);

}

localStorage.setItem('tasks', JSON.stringify(tasks));

}

</script>

</body>

</html>

В ходе технологической практики было разработано веб–приложение согласно требованиям, изложенным в главе 5.1. Однако, для того чтобы разработку можно было считать завершенной, необходимо провести тестирование. Тестирование программного обеспечения – проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов, выбранном определенным образом. Главная цель проведения тестирование – проверка выполнения программой поставленных задач, способность решать задачи пользователей с необходимой точностью при использовании в заданном контексте, способность предоставлять определенные результаты в рамках ожидаемых затрат ресурсов и способность приносить удовлетворение пользователям при использовании в заданном контексте. Для проверки выполнения программой всех вышеуказанных требований, необходимо провести ряд тестов, среди которых можно выделить следующие:

– Smoke – тесты – подразумевает минимальное покрытие и проверка работы основных функций.

– Функциональное тестирование – устанавливает соответствие разработанного программного обеспечения исходным функциональным требованиям к программному средству.

– GUI – подразумевает проверку единство дизайна, орфографии, дружественность и т.д.

В первую очередь было проведено Smoke – тестирование. Результаты тестирования представлены в таблице 5.1.

Таблица 5.1 – Результаты Smoke – тестов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Приоритет | Уровень тестового покрытия | Тест | Результат |
| 1 | Наивысший | Smoke | Запуск интерфейса приложения | Тест  пройден |
| 2 | Наивысший | Smoke | Добавление данных | Тест  пройден |
| 3 | Наивысший | Smoke | Удаление данных | Тест  пройден |
| 4 | Наивысший | Smoke | Завершение данных | Тест  пройден |
| 5 | Средний | Smoke | Сортировка данных | Тест  пройден |

Следующий шаг тестирования программного обеспечения – проведения функционального тестирования. Функциональное тестирование является одним из ключевых видов тестирования, задача которого – установить соответствие разработанного программного обеспечения исходным функциональным требованиям заказчика. То есть проведение функционального тестирования позволяет проверить способность информационной системы в определенных условиях решать задачи, согласно предъявленным требованиям. Результаты тестирование разработанной программы представлены ниже в таблице 5.2.

Таблица 5.2 – Результаты функционального тестирования

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название проверок | Ожидаемый результат | Полученный результат |
| 1 | Добавление задачи | Корректное добавление | Тест пройден |
| 2 | Завершение задачи | Корректное завершение | Тест пройден |
| 3 | Сортировка задачи | Корректная сортировка | Тест пройден |
| 4 | Удаление задачи | Корректное удаление | Тест пройден |

Далее требуется провести тестирование пользовательского интерфейса (UI). Результаты тестирования программы представлены в таблице 5.3.

Таблица 5.3 – Результаты тестирование пользовательского интерфейса

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название  проверок | Ожидаемый результат | Полученный результат |
| 1 | Единство дизайна | Одинаковое оформление всех панелей приложения | Тест  пройден |
| 2 | Орфография | Содержание соответствует орфографическим и грамматическим аспектам языка | Тест  пройден |
| 3 | Присутствие  элементов | Необходимые элементы управления отображаются корректно и находятся на своих местах | Тест  пройден |
| 4 | Дружественность | Программа хорошо воспринимается и понятна | Тест  пройден |
| 5 | Скорость работы | Программа быстро загружается и не создает неудобств | Тест  пройден |

Подводя итог, требуется отметить, что программное обеспечение стабильно работает и имеет интуитивно понятный и простой интерфейс. Скорость работы программы достаточно быстрая, в нем легко ориентироваться. Все команды работают без ошибок. Качество данного приложения находится на высоком уровне. Дефекты функционала не обнаружены. Таким образом, разработанная программа соответствует всем заявленным характеристикам.

# Заключение

По итогам прохождения производственной практики была изучена структура организации ООО «Andersen», а именно, произведен анализ характеристики предприятия, рассмотрена его организационная структура, произведен анализ функционирования структурных подразделений предприятия. Были изучены нормативные документации подразделения, в том числе должностная инструкция инженера-программиста. В результате выполнения индивидуального задания, были изучены и закреплены знания HTML, JavaScript, CSS.

Технологическая (производственная) практика сыграла огромную роль в приобретении практических навыков. В ходе неё были закреплены теоретические знания по изученным дисциплинам и было получено много новых.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Электронный ресурс «JavaScript – чтение данных» [Электронный ресурс]. Режим доступа https://coderlessons.com/tutorials/veb-razrabotka/izuchite-firebase/firebase-chtenie-dannykh. Дата доступа – 12.08.2023 г.
2. CardView (Карточка)– Режим доступа: http://developer.alexanderklimov.ru/android/views/cardview.php. Дата доступа: 12.08.2023 г.
3. Официальная документация mybatis [Электронный ресурс]. Режим доступа https://mybatis.org/mybatis-3/. Дата доступа – 12.08.2023 г

# ПРИЛОЖЕНИЕ А

**(обязательное)**

**А.1 Назначение программы**

Назначение приложения - упростить управление задачами и делами, помочь пользователям структурировать свою деятельность и повысить их продуктивность, обеспечивая легкий доступ к спискам дел в любое время и в любом месте.

**А.2 Условия выполнения программы**

Для правильной и стабильной работы созданной системы, компьютер, на котором будет использоваться созданная программа, должен обладать следующими минимальными характеристиками:

* 4012 MB RAM;
* свободное место на жестком диске 1500 МБ;
* устройства ввода/вывода.

На ПК должна быть установлена любая операционная система, поддерживающая установку любого современного браузера.

**А.3 - Выполнение программы**

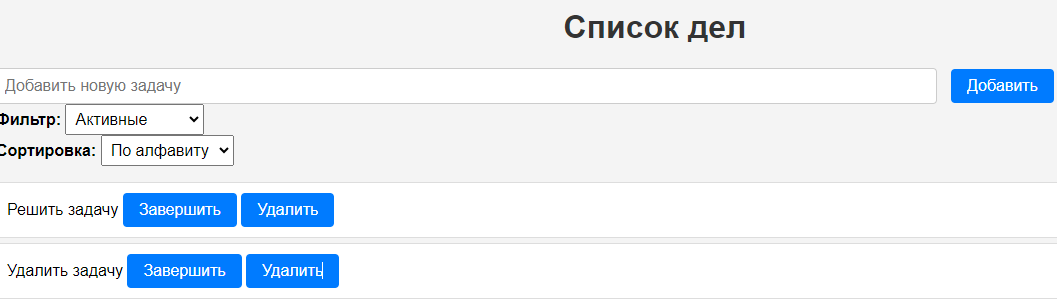
****

Рисунок А.1 – Активные задачи

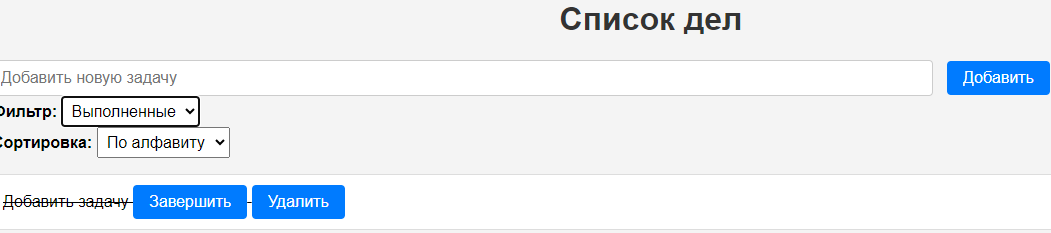


Рисунок А.2 – Выполненные задачи