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 7](#_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 23](#_Toc145008172)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 24](#_Toc145008173)

[ПРИЛОЖЕНИЕ А 25](#_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 To Do List. Это приложение предоставит пользователям эффективный инструмент для управления и организации их задач, дел и списка дел.

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 — международная IT-компания, которая специализируется на разработке программного обеспечения, предоставлении консультационных услуг и цифровой трансформации. Компания Andersen была основана в 2007 году и в настоящее время имеет офисы и представительства в различных странах. История развития компании Andersen связана с ее стремлением к постоянному инновационному росту и внедрению новых технологий. Компания активно занималась разработкой программного обеспечения, предоставлением услуг по тестированию и контролю качества, а также консалтинговыми услугами.

Со временем, Andersen стала шире внедрять искусственный интеллект, автоматизацию бизнес-процессов, машинное обучение и другие передовые технологии в свои проекты. Компания активно работает с клиентами из различных отраслей, включая финансы, здравоохранение, розничную торговлю, производство и другие, помогая им добиться цифровой трансформации и достичь конкурентного преимущества на рынке.

Сегодня Andersen является одной из ведущих IT-компаний, предоставляющих техническую экспертизу и консультирование в области разработки ПО и цифровой трансформации. Компания постоянно развивается, и их целью является опережающее предоставление инновационных решений для своих клиентов.

## Организационная структура предприятия

В настоящее время штат постоянно растет и на данный момент насчитывает более 3500 сотрудников, а каждый из новых специалистов принес в команду свой уникальный опыт и наработки. Около 70% сотрудников Andersen заняты непосредственно разработкой, а еще 15% - тестированием. Работа компании построена в полном соответствии с методологией Agile и практикой непрерывной интеграции. Это означает, что каждый член команды всегда знает свои обязанности, они не пересекаются с обязанностями других сотрудников и не мешают их выполнению.

Благодаря собственному отделу тестирования обеспечивается неизменно высокое качество разрабатываемых решений. Кроме того, предлагается тестирование сторонних проектов как отдельная услуга.

В активе Andersen есть проекты различных типов: разработка приложений для предприятий госсектора, внедрение электронных платежных систем, ряд приложений для мобильных устройств и многие другие.

Основные задачи компании:

* Разработка приложений от настольных до веб, работа с торговой и производственной сферами, а также со сферой рекламы и развлечений. Интегрирование разнородного ПО. Сведение различных информационных систем в единую платформу.
* Проведение в плановом и экстренном режиме проверки работоспособности web-ресурсов, приложений и электронных игр, где Andersen выступает как независимый эксперт.
* Создание стильных, красочных, адаптивных и мобильных решений с удобным интерфейсом и уникальным дизайном.
* Настройка и доработка типовых конфигураций 1С. Разработка новых отчетов и обработок 1С. Разработка конфигураций с нуля под заказ.[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

Одно из основных направлений деятельности компании – разработка программного обеспечения на заказ (аутсорсинг). Типичный заказчик Andersen обращается к в компанию, когда для его задач не существует достаточно удобного и функционально соответствующего готового универсального решения. Команда Andersen полностью берет процесс под свой контроль. Создание решения начинается с четкого формулирования требований и анализа потребностей клиента. Далее обязательно следует создание визуальных моделей и предоставление их заказчику на согласование. Специалисты Andersen всегда активно участвуют в процессе согласования и дают клиенту экспертные рекомендации.

Разработка проекта ведется с использованием безопасной и гибкой методики Agile. При планировании разработки используется декомпозиция с оптимальным размером фрагментов. Методология непрерывной интеграции позволяет управлять рисками, что существенно ускоряет выход продукта. Эффективные коммуникации и отслеживание ошибок обеспечиваются использованием системы JIRA, а хранение всех промежуточных версий кода – применением Git. Для различных уровней (разработка, продакшн, поддержка, тестирование) применяются отдельные среды, что исключает возникновение проблем и обеспечивает поступление заказчику исключительно рабочих сборок ПО.

Для тестирования разрабатываемого программного обеспечения в Andersen выделен специальный отдел, что позволяет обеспечить не только полное соответствие разрабатываемого решения требованиям технического задания, но и снизить риски, связанные с его поведением в нештатных ситуациях.

# 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.
* 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.

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 отслеживания повседневных задач.

## 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;

– фильтрация задач;

– сортировка задач;

– интерфейс;

– обработка событий.

## 5.5 Реализация и тестирование программы

Для реализации интерактивного приложения потребовались HTML, CSS и JavaScript. Код файла index.html представлен в листинге 5.1.

Листинг 5.1 – Реализация веб-сайта

<!DOCTYPE html>

<html lang="ru">

<head>

    <link rel="stylesheet" href="./styles/normalize.css">

    <link rel="preconnect" href="https://fonts.googleapis.com">

    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>

    <link

        href="https://fonts.googleapis.com/css2?family=Montserrat:wght@100;200;300;400;500;600;700;800;900&display=swap"

        rel="stylesheet">

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="./styles/style.css">

    <script defer type="module" src="./script/main.js"></script>

    <title>ToDo List</title>

</head>

<body>

    <div class="container">

        <div class="task-container">

            <h1 class="header">HIGH</h1>

            <form class="task-container\_\_input-container high-task">

                <input id="input-high" class="task-container\_\_input" type="text" placeholder="Добавить важные дела" onkeypress="addTask(event, 'high-task')">

                <button id="button-high-add-task" class="task-container\_\_button"></button>

            </form>

            <div class="task">

Продолжение листинга 5.1

<input id="change-box" class="radio-button" type="checkbox">

                <p for="change-box" class="task-text">Приготовить обед.</p>

                <button id="button-delete" class="task-container\_\_button task-container\_\_button\_45degrees"></button>

            </div>

            <div class="task">

                <input id="change-box" class="radio-button" type="checkbox">

                <p for="change-box" class="task-text">Сверстать этот TODO list.</p>

                <button id="button-delete" class="task-container\_\_button task-container\_\_button\_45degrees"></button>

            </div>  </div>

        <div class="task-container">

            <h1 class="header">LOW</h1>

            <form class="task-container\_\_input-container low-task">

                <input id="input-low" class="task-container\_\_input" type="text" placeholder="Добавить" onkeypress="addTask(event, 'low-task')">

                <button id="button-low-add-task" class="task-container\_\_button"></button>

            </form>

            <div class="task">

                <input id="change-box" class="radio-button" type="checkbox">

                <p for="change-box" class="task-text">Посмотреть ютубчик.</p>

                <button id="button-delete" class="task-container\_\_button task-container\_\_button\_45degrees"></button>

            </div>

        </div>

    </div>

</body>

</html>

Для визуального оформления использовали CSS. Код файла style.css представлен в листинге 5.2.

Листинг 5.2 – Реализация веб-сайта

\*{

box-sizing: border-box;

}

p

{

margin: 0px;

margin-top: 3px;

}

Html

{

font-family: 'Montserrat', sans-serif;

Продолжение листинга 5.2

}

button:active, button:focus {

outline: none;

}

button::-moz-focus-inner {

border: 0;

}

.container{

border: 1px solid rgb(139, 139, 139);

border-radius: 25px;

width: 500px;

margin: 10px auto 0 auto;

padding: 5px 50px 38px 50px;

}

.task-container

{

display: flex;

align-items: center;

flex-direction: column;

width: 100%;

}

.header{

font-size: 17px;

font-weight: 500;

letter-spacing: -1px;

padding-top: 10px;

}

.task-container\_\_input-container

{

display: flex;

align-items: center;

width: 100%;

height:43px;

justify-content: space-between;

padding:0px 10px 0px 10px;

border: 1px solid rgb(139, 139, 139);

border-radius: 6px;

transition: 0.3s;

}

.task-container\_\_input

{

border: none;

outline: none;

color: rgba(69, 69, 69, 0.819);

font-weight: 500;

width: 100%;

height: 30px;

padding-right: 20px;

line-height: 1;

}

Продолжение листинга 5.2

.task-container\_\_input-container:hover

{

transition: 0.4s;

transform: scale(1.015);

}

.task-container\_\_button{

transition: 0.3s;

background: none;

border: none;

cursor: pointer;

background: url(/img/plus.svg) center no-repeat;

width: 20px;

height: 20px;

}

.task-container\_\_button:hover {

transition: 0.5s;

background: url(/img/plus-green.svg) center no-repeat;

transform: scale(1.1);

}

.task-container\_\_button\_45degrees{

transition: 0.4s;

transform: rotate(45deg);

}

.task-container\_\_button\_45degrees:hover{

transition: 0.3s;

background: url(/img/plus-red.svg) center no-repeat;

transform: scale(1.1) rotate(45deg);

}

.task

{

display: flex;

align-items: flex-start;

justify-content: space-between;

margin-top: 10px;

padding:10px 10px 10px 10px;

border: 1px solid rgb(139, 139, 139);

border-radius: 6px;

width: 100%;

height: 100%;

transition: 0.4s;

}

.task:hover{

transform: scale(1.02);

}

.radio-button

{

transform:scale(1.3);

transition: 0.4s;

margin-top: 3.5px;

cursor: pointer;

Продолжение листинга 5.2

appearance: none;

width: 13px;

height: 13px;

border: 0.8px solid rgb(139, 139, 139);

border-radius: 50%;

}

.radio-button:after {

content: '';

display: block;

width: 10px;

height: 10px;

background: url(/img/check.svg) no-repeat;

background-size: 8px 9px;

opacity: 0;

transition: 1s;

position: absolute;

top: 1.7px;

left: 1.8px;

}

.radio-button:checked {

background-color: rgb(153, 153, 153);

}

.radio-button:checked::after {

opacity: 1;

transition: opacity 1s;

}

.radio-button:hover {

transform:scale(1.4);

transition: 0.4s;

}

input[type='checkbox'] {

accent-color: #6e6e6e;

}

.task-text

{

hyphens: auto;

word-wrap: break-word;

font-weight: 400;

font-size: 2;

line-height: 1;

width: 87%;

}

.task:has(.radio-button:checked)

{

background-color: rgb(210, 210, 210);

transition: 1s;

transform: scale(1);

}

Для добавления интерактивности динамического контента использовали JavaScript. Код файла main.js представлен в листинге 5.3.

Листинг 5.3 – Реализация веб-сайта

const form = document.querySelector('#addTask');

const input = document.querySelector('#taskName');

const input2 = document.querySelector('#taskName2');

input.addEventListener('keydown', (event) => {

if (event.key === 'Enter') {

event.preventDefault();

const highPriorityTask = document.querySelector('#high-priority #tasks');

const taskName = input.value;

input.value = '';

const newTask = document.createElement('div');

newTask.setAttribute('id', 'task');

newTask.innerHTML = `

<form action="">

<label>

<input type="checkbox">

<span class="checkmark"></span>

<p>

${taskName}

</p>

</label>

<button id="deleteButton">x</button>

</form>

`;

highPriorityTask.appendChild(newTask);

}

});

input2.addEventListener('keydown', (event) => {

if (event.key === 'Enter') {

event.preventDefault();

const lowPriorityTask = document.querySelector('#low-priority #tasks');

const taskName = input2.value;

input2.value = '';

const newTask = document.createElement('div');

newTask.setAttribute('id', 'task');

newTask.innerHTML = `

<form action="">

<label>

<input type="checkbox">

<span class="checkmark"></span>

<p>

${taskName}

Продолжение листинга 5.3

</p>

</label>

<button id="deleteButton">x</button>

</form>`;

lowPriorityTask.appendChild(newTask);

}

});

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 | Завершение данных | Тест  пройден |
| 4 | 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 |

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 - структурировать рабочий процесс, улучшить планирование и увеличить продуктивность.

**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**

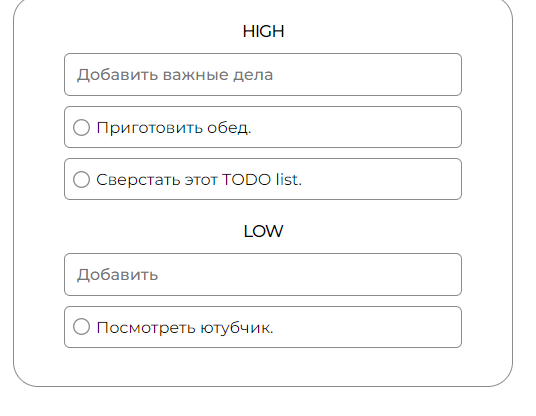
****

Рисунок А.1 – Активные задачи

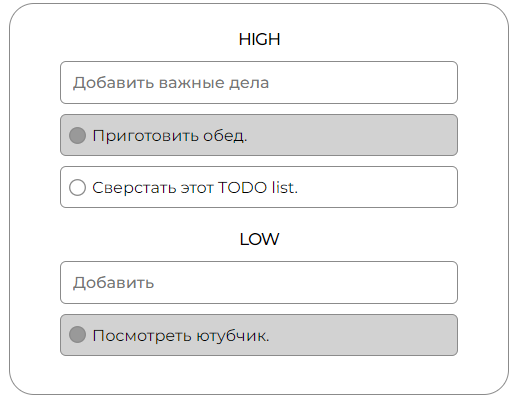


Рисунок А.2 – Выполненные задачи