Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет информационных технологий и управления

Кафедра информационных технологий автоматизированных систем

|  |
| --- |
| *К защите допустить*: |
| Заведующий кафедрой ИТАС |
| \_\_\_\_\_\_\_\_\_\_\_ А.А. Навроцкий |

Пояснительная записка

к дипломному проекту

на тему

**АВТОМАТИЗИРОВАННАЯ СИСТЕМА УПРАВЛЕНИЯ ПОЛОЧНЫМ ПРОСТРАНСТВОМ СУПЕРМАРКЕТА**

БГУИР ДП 1-53 01 02 06 013 ПЗ

|  |  |  |
| --- | --- | --- |
| Студент |  | В. Г. Колесников |
| Руководитель |  | В. Л. Арановский |
| Консультанты: |  |  |
| *от кафедры ИТАС* |  | Т. С. Боброва |
| *по экономической части* |  | Т. Л. Слюсарь |
| Нормоконтролер |  | В. И. Ярмолик |
| Рецензент |  |  |

Минск 2020

**РЕФЕРАТ**

АВТОМАТИЗИРОВАННАЯ СИСТЕМА УПРАВЛЕНИЯ ПОЛОЧНЫМ ПРОСТРАНСТВОМ СУПЕРМАРКЕТА: дипломный проект / В. Г. Колесников. – Минск : БГУИР, 2020, – п.з. – 64 с., чертежей (плакатов) – 6 л. формата А1.

Цель работы: разработка автоматизированной системы управления полочным пространством супермаркета.

Пояснительная записка к дипломному проекту состоит из введения, 4 разделов, включающих характеристику объекта разработки, разработку автоматизированной системы управления полочным пространством супермаркета, программную реализацию автоматизированной системы, а также технико-экономическое обоснование разработки системы, заключение, список использованных источников.

Для разработки программного комплекса был выбран язык программирования Python. Система представляет собой клиентское приложение, написанное с использованием библиотеки React, и серверное приложение, написанное на фреймворке Django. Доступ к приложению пользователи будут получать посредством веб-браузера.

Результатом работы стала автоматизированная система управления полочным пространством супермаркета, предоставляющая возможность собирать статистику об изменениях артериального давления, отслеживать их динамику, получать оповещения и рекомендации о возможных изменениях артериального давления. Программный комплекс также включает в себя базу данных, которая будет формироваться в ходе работы приложения, и может иметь большое значение в сфере здравоохранения, как агрегатор показателей артериального давления различных категорий людей при различных условиях.

Министерство образования Республики Беларусь

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Учреждение образования БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | |
| Факультет | | | ИТиУ | | | | Кафедра | | | | ИТАС | | | | | | | | | | | |
| Специальность | | | 1-53 01 02 | | | | Специализация | | | | | | | 06 | | | | | | | | |
| УТВЕРЖДАЮ | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | |  | | | | | Зав. кафедрой | | | | |
| « | | | | | | | | | | | | | 2 | | » | | апреля | | | 2020 | г. | |
| **ЗАДАНИЕ** | | | | | | | | | | | | | | | | | | | | | | |
| **по дипломному проекту (работе) студента** | | | | | | | | | | | | | | | | | | | | | | |
| Колесникова Владислава Гавриловича | | | | | | | | | | | | | | | | | | | | | | |
| (фамилия, имя, отчество) | | | | | | | | | | | | | | | | | | | | | | |
| 1 Тема проекта (работы): | | | | Автоматизированная система управления полочным | | | | | | | | | | | | | | | | | | |
| пространством супермаркета | | | | | | | | | | | | | | | | | | | | | | |
| утверждена приказом по университету от | | | | | | | « | 01 | | » | | апреля | | | | 2020 г. | | | № | 750-с | | |
| 2 Срок сдачи студентом законченной работы | | | | | | | | | 01.06.2020 | | | | | | | | | | | | | |
| 3 Исходные данные к проекту: | | | | | версия операционной системы – Windows 7 и выше; | | | | | | | | | | | | | | | | | |
| СУБД – *mongodb*; | | | | | | | | | | | | | | | | | | | | | | |
| язык программирования – *Python*; Библиотека – *React*; Фреймворк – *Django*; | | | | | | | | | | | | | | | | | | | | | | |
| библиотека - Express; | | | | | | | | | | | | | | | | | | | | | | |
| Назначение разработки: | | | | | | | | | | | | | | | | | | | | | | |
| разработать автоматизированную систему управления полочным пространством | | | | | | | | | | | | | | | | | | | | | | |
| супермаркета | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | |
| 4 Содержание пояснительной записки (перечень подлежащих разработке вопросов): | | | | | | | | | | | | | | | | | | | | | | |
| Введение | | | | | | | | | | | | | | | | | | | | | | |
| 1 Анализ объекта автоматизации | | | | | | | | | | | | | | | | | | | | | | |
| 2 Проектирование структуры автоматизированной системы управления полочным | | | | | | | | | | | | | | | | | | | | | | |
| пространством супермаркета | | | | | | | | | | | | | | | | | | | | | | |
| 3 Программная реализация автоматизированной системы | | | | | | | | | | | | | | | | | | | | | | |
| 4 Технико-экономическое обоснование разработки автоматизированной системы | | | | | | | | | | | | | | | | | | | | | | |
| Заключение | | | | | | | | | | | | | | | | | | | | | | |
| 5 Перечень графического материала (с точным указанием обязательных чертежей): | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | |
| 6 Содержание задания по технико-экономическому обоснованию.  Технико-экономическое обснование разработки и использования автоматизированной | | | | | | | | | | | | | | | | | | | | | | |
| системы управления полочным пространством супермаркета | | | | | | | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | | | | | | | |
| Задание выдал |  | | | | Т. Л. Слюсарь | | | | | | | | | | | | | | | |

КАЛЕНДАРНЫЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапов дипломного проекта (работы) | Объём этапа, % | Срок выполнения этапа | Примечание |
| Сбор и изучение материалов по теме дипломного | 40 | 20.04.2020 |  |
| проектирования. Написание раздела 1 пояснительной записки, расчет технико-экономического обоснования |
| Проектирование программного комплекса. Написание раздела 2 пояснительной записки и соответствующего графического материала | 60 | 04.05.2020 |  |
| Реализация программного комплекса. Написание раздела 3 пояснительной записки и соответствующего графического материала | 80 | 18.05.2020 |  |
| Оформление пояснительной записки и подготовка презентации | 100 | 01.06.2020 |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Дата выдачи задания | 02.04.2020 | | Руководитель | |  | Т.С. Боброва |
| Задание принял к исполнению | |  | | В.Г. Колесников | |

**СОДЕРЖАНИЕ**

[Перечень условных обозначений, символов и терминов 5](#_Toc9905439)

[Введение 6](#_Toc9905440)

[1 Характеристика объекта разработки 8](#_Toc9905441)

[1.1 Система кровообращения человека 8](#_Toc9905442)

[1.2 Анализ значений артериального давления 11](#_Toc9905444)

[1.3 Анализ существующих аналогов разрабатываемого программного комплекса 12](#_Toc9905445)

[1.4 Постановка задачи 17](#_Toc9905449)

[2 Проектирование структуры программного комплекса анализа артериального давления 18](#_Toc9905450)

[2.1 Структура разрабатываемого программного комплекса 18](#_Toc9905451)

[2.2 Проектирование программного комплекса с использованием инструментальных средств 19](#_Toc9905452)

[2.3 Информационное обеспечение программного комплекса 25](#_Toc9905455)

[2.4 Алгоритмическое обеспечение 28](#_Toc9905459)

[2.5 Техническое и системное программное обеспечение 33](#_Toc9905460)

[2.6 Эргономическое обеспечение 33](#_Toc9905461)

[3 Программная реализация программного комплекса анализа артериального давления 35](#_Toc9905462)

[3.1 Обоснование выбора средств разработки 35](#_Toc9905463)

[3.2 Структура программного продукта 38](#_Toc9905464)

[3.3 Проектирование программной реализации комплекса 38](#_Toc9905465)

[3.4 Описание программной реализации отдельных частей комплекса 39](#_Toc9905466)

[3.5 Руководство пользователя 54](#_Toc9905473)

[4 Технико-экономическое обснование разработки программного комплекса анализа артериального давления 55](#_Toc9905474)

[4.1 Характеристика программного продукта 55](#_Toc9905475)

[4.2 Оценка трудоемкости и сроков разработки 55](#_Toc9905476)

[4.3 Расчет затрат на разработку программного комплекса 58](#_Toc9905477)

[4.4 Расчёт экономической эффективности реализации программного комплекса аналиха артериального давления 60](#_Toc9905478)

[4.5 Результат технико-экономического обоснования разработки программного продукта 61](#_Toc9905479)

[Заключение 62](#_Toc9905480)

[Список использованных источников 63](#_Toc9905481)

[Приложение А (обязательное) Листинг моделей соединения с базой данных 64](#_Toc9905482)

[Ведомость документов дипломного проекта 6](#_Toc9905483)5

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ, символов и терминов

Торговый объект – имущественный комплекс, а также иное имущество, используемое для осуществления торговли, принадлежащие на праве собственности, праве хозяйственного ведения, праве оперативного управления или на ином законном основании торговым организациям и (или) индивидуальным предпринимателям.

Магазин – специально оборудованное стационарное здание или его часть, предназначенное для продажи товаров и оказания услуг покупателям и обеспеченное торговыми, подсобными, административно-бытовыми помещениями, а также помещениями для приема, хранения и подготовки товаров к продаже.

Киоск – оснащенное торговым оборудованием строение, не имеющее торгового зала и помещений для хранения товаров, рассчитанное на одно рабочее место продавца, на площади которого хранится товарный запас.

Универсальный магазин – предприятие розничной торговли, реализующее универсальный ассортимент продовольственных и непродовольственных товаров.

Супермаркет – магазин с торговой площадью от 650 до 4000 квадратных метров, в котором реализуется универсальный ассортимент продовольственных товаров и ограниченный ассортимент непродовольственных товаров методами самообслуживания, традиционного обслуживания, продажи товаров по предварительным заказам.

Гипермаркет – предприятие торговли, реализущее продовольственные и непродовольственные товары универсального ассортимента преимущественно по форме самообслуживания.

Планограмма – документ, в котором детально изображается выкладка товаров с точным указанием мест размещения на торговом оборудовании торгового предприятия ассортиментных позиций.

Мерчендайзинг – оптимизация системы торговли, связанная с подготовкой товаров, их рекламой, а также стимулированием торговой деятельности.

*UML (Unified Modeling Language) –*унифицированный язык объектно-ориентированного моделирования.

*IDEF*0 – методология, которая представляет собой совокупность методов, правил и процедур, предназначенных для построения функциональной модели объекта какой-либо предметной области.

***IDEF*3** – методология моделирования, использующая графическое описание информационных потоков, взаимоотношений между процессами обработки информации и объектов, являющихся частью этих процессов.

СУБД – совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

# ВВЕДЕНИЕ

В связи с постоянным развитием и ростом сферы торговли, а также с постоянным улучшением процессов автоматизации, рано или поздно эти компоненты должны связаться между собой, образовав композицию из сложных систем, взаимодействующих между собой для минимизации расходов на шаблонные операции и максимизации прибыли торгового предприятия.

Примером шаблонной операции может являться работа с полочным пространством супермаркета. Данный процесс включает в себя не только расстановку товаров на полках, но и контроль за их количеством, а также их правильной расстановкой. Если в небольшом торговом объекте с этим не возникает больших проблем, то с увеличением размеров объекта появляются сложности, которые требуют увеличения торгового персонала, что вызывает рост расходов. Довольно крупные предприятия изначально планируют систему, которая позволит минимизировать влияние этих процессов на работу торгового объекта. Однако для торговых объектов среднего размера иногда данный фактор не учитывается в полной мере.

Внедрение в торговый процесс разработок, направленных на облегчение выполнения повседневных и многоразовых операций, позволяет многократно сократить затраты на те или иные действия при разумных вложениях.

Одним из вариантов частичной автоматизации работы супермаркета может являться автоматизированная система управления полочным пространством. Прежде всего, данная система позволяет следить за общим состоянием магазина в целом и каждой полки по отдельности в режиме реального времени на основе производимых операций купли-продажи и выкладки товаров на полки. Кроме того, данная система позволяет собирать и анализировать данные, поступающие в процессе работы супермаркета, на основе чего, вырабатывается дальнейшая тактика работы с выдачей определенных рекомендаций.

Для разработки данной системы необходимо решить следующие задачи:

* исследовать преимущества и недостатки существующих автоматизированных систем такого же или похожего типа;
* определить требования к автоматизированной системе;
* спроектировать автоматизированную систему;
* экономически либо социально обосновать разработку и реализовать автоматизированную систему.

Преимуществом данной автоматизированной системы является база данных, которая будет заполняться данными о товарах, их количестве на полках и складе, а также точным или приблизительным сроком годности. Кроме того, система обладает возможностью анализировать ежедневные операции с товарами, на основе чего строится статистика работы супермаркета, и настраиваются показатели на очередной день его работы.

Тема дипломного проекта: «Автоматизированная система управления полочным пространством супермаркета». Целью данного дипломного проекта является разработка и реализация автоматизированной системы управления полочным пространством супермаркета. Система позволит следить за общим состоянием магазина и каждой полки по отдельности, а также выдавать статистические данные о работе магазина с последующей настройкой параметров, что позволит сократить расходы на персонал.

Дипломный проект выполнен самостоятельно, проверен в системе «Антиплагиат». Процент оригинальности соответствует норме, установленной кафедрой (68%). Цитирования обозначены ссылками на публикации, которые указанны в разделе «Список использованных источников».

# АНАЛИЗ ОБЪЕКТА АВТОМАТИЗАЦИИ

## Описание объекта автоматизации

Принято различать торговые объекты по их видам и типам, которые, в свою очередь, зависят от различных факторов. Существует большое количество разновидностей магазинов. В основе их классификации лежат следующие признаки:

* тип ассортимента;
* уровень цен;
* форма обслуживания потребителей;
* местонахождение.

Каждый из них оказывает существенное влияние на прибыль бизнеса, но, в зависимости, например, от региона, экономической ситуации в данный период времени и прочих субъективных факторов, содержит те или иные признаки [1].

Большое влияние на тип торгового объекта оказывает его местоположение. Например, в таких местах, как остановки общественного транспорта, наиболее распространены киоски. Бóльшим разнообразием отличаются места, где чаще всего можно заметить массовое скопление людей. В таких местах обычно располагаются торговые центры, которые могут иметь как гигантские универсальные магазины, так и маленькие продуктовые магазины. Также существует такой класс гипермаркетов, которые могут располагаться за пределами города или на его границе. Обычно, такая форма торгового объекта предлагает широкий ассортимент товаров разнообразных классов и видов. Предназначены такие гипермаркеты преимущественно для людей, которые посещают магазин раз в несколько недель, а иногда и реже.

На данный момент размер торгового объекта и его ассортимент определяется, прежде всего, наличием спроса на тот или иной товар. Так, если спрос на строительные материалы преобладает в определенной местности, вероятнее всего, там может появиться специализированный магазин, который будет удовлетворять нужды покупателей.

Исходя из спроса, чаще всего происходит регулирование цен на ту или иную продукцию. Так, например, при достаточном количестве товара в разных торговых объектах и одинаковом спросе на него, при условии, что эти торговые объекты находятся неподалеку друг от друга, выиграет тот магазин, у которого цена на товар будет более привлекательной, нежели в другом. К сожалению, несмотря на регулирование цен, существуют ситуации, в которых цена на товар может варьироваться в зависимости от предложения данного товара. Так, например, при дефиците товара определенного вида его цена будет большей, чем он мог бы стоить на самом деле. Чаще всего такие ситуации происходят в малых городах и деревнях, где товары сами по себе являются дефицитными в таких местах.

При достаточном количестве торговых объектов в больших городах иногда выигрывает тот, который является более дружелюбным и подходящим для клиента. То есть если магазины предлагают определенный товар по примерно одинаковой цене, может выиграть тот, в котором, по мнению покупателя, персонал более снисходителен и внимателен к покупателю, даже если цена в этом торговом объекте немного больше, чем в другом. Кроме того, если в магазине товары расположены таким образом, что найти нужную позицию бывает довольно сложно, а иногда и невозможно, либо торговое пространство распределено крайне неграмотно, такой магазин может потерять клиента в связи с данными обстоятельствами.

Учитывая выделенные факторы, необходимо проанализировать каждый вид торгового объекта, чтобы выделить тот вид, которому действительно будет необходима проектируемая автоматизированная система.

1. **Розничный магазин**

В данном случае розничный магазин рассматривается как торговый объект, имеющий достаточно пространства для нахождения в нем от одного до нескольких продавцов и покупателей.

Розничные магазины имеют свои особенности по сравнению с универсальными магазинами: в небольших магазинах покупатель сразу же видит весь ассортимент товаров. Внутренний вид продовольственного магазина представлен на рисунке 1.1.

Также, кроме своих размеров, розничный магазин может специализироваться на предоставляемых услугах. Например, существуют специализированные магазины, предоставляющие товары определенного вида или класса, такие как магазины стройматериалов, ювелирные магазины, магазины бытовой техники, магазины косметики и другие. Внутренний вид магазина стройматериалов представлен на рисунке 1.2.

Так как в магазине могут присутствовать лишь несколько продавцов и покупателей, а ассортимент товаров зачастую представляет собой небольшую группу, относительно универсальных магазинов, данный вид торговых объектов является довольно простым во взаимодействии между покупателями и продавцами.



Рисунок 1.1 – Продовольственный магазин



Рисунок 1.2 – Магазин строительных материалов

Проанализировав выделенные преимущества и недостатки данного вида торговых объектов, можно прийти к выводу, что небольшие магазины могут нуждаться только в автоматизации составления планограмм. Глубокий и расширенный анализ продаж и оптимизации расположения и количества товаров на полках магазина не требуются, так как расстановкой товаров на полках и их правильным расположением занимается продавец в течение рабочего дня.

1. **Супермаркет**

Супермаркет является одним из видов универсальных магазинов. В данном виде торговых объектов присутствует большой ассортимент продовольственных товаров и относительно небольшой ассортимент непродовольственных товаров, что уже дает ему преимущество перед розничными магазинами и киосками. Внутренний вид супермаркета представлен на рисунке 1.3.



Рисунок 1.3 – Супермаркет

Для работы целого супермаркета чаще всего нанимается персонал, представляющий собой группу специально обученных людей, состоящую из начального, производственного, среднего уровней и топ-менеджмента [2].

В супермаркете присутствуют большие стеллажи и длинные полки с продуктами. Нехватка продуктов на полках может являться как следствием недостаточного количества товаров на складе, так и следствием человеческого фактора, в результате которого продукты не расставлены на полках из-за невнимательности или нехватки времени.

В период акций и предложений, контроль за продукцией должен повышаться, особенно – за акционной продукцией. Кроме того, работа супермаркета в разное время дня, как и в разное время года, будет отличаться. Это тоже стоит учитывать при планировании нагрузки персонала.

Выделив определенные достоинства и недостатки данного вида торговых объектов, можно сделать вывод, что супермаркетам может понадобиться автоматизированная система управления полочным пространством, которая будет анализировать поступающие данные о работе супермаркета, на основе чего можно будет оптимизировать его работу.

1. **Гипермаркет**

Гипермаркет, как и супермаркет, является универсальным магазином. Гипермаркеты намного больше супермаркетов. Из-за своих больших размеров, гипермаркет может сам по себе являться складом. Кроме больших размеров, гипермаркеты выделяются тем, что могут предоставить богатый ассортимент товаров разного вида, как продовольственных, так и непродовольственных. Внешний вид гипермаркета изображен на рисунке 1.4.



Рисунок 1.4 – Гипермаркет

Данный вид торговых объектов весьма сложен в управлении, поэтому гипермаркеты набирают в менеджеры людей с опытом работы в данной сфере. Гипермаркеты являются очень большим комплексом, требующим тщательного и грамотного контроля за всеми его частями. Специфика гипермаркета подразумевает высокую степень механизации и автоматизации всех работ, связанных как со внутренней системой магазина, так и с логистикой [3].

Выделенный набор особенностей гипермаркета показывает, что данная автоматизированная система будет недостаточной для управления данным видом торгового объекта. Для контроля гипермаркета необходимо разрабатывать целый комплекс взаимодействующих между собой систем, что выходит за рамки разрабатываемого дипломного проекта.

## Анализ современных методов автоматизации работы супермаркетов

Современные методы автоматизации предоставляют возможность автоматизировать многие процессы работы магазинов. Системы и средства позволяют внедрить свои компоненты в уже существующие информационные технологии, которые используются в супермаркетах. Некоторые решения требуют только программного обеспечения, но есть и такие системы, для работы которых необходимо еще и аппаратное обеспечение. Кроме того, популярность набирают облачные решения, предоставляющие функционал даже без дополнительных закупок или перенастроек оборудования.

## Программные решения

Многие компании изначально предоставляли большое разнообразие программных продуктов, поставляемых прямиком на компьютер заказчика. Данные решения позволяли выполнять именно заказанные задачи прямиком “из коробки”, то есть они предоставляли тот функционал, который был нужен заказчику, без дополнительных установок другого программного обеспечения.

Такие программные продукты пользуются популярностью по сей день благодаря простоте их приобретения и использования, хотя в них и есть явные недостатки, которые могут повлиять на дальнейшее развитие предприятия. Так, например, при заказе программного продукта, могут сложиться трудности при его установке и сопровождении, так как иногда полученный продукт может конфликтовать с другими предустановленными программными продуктами. Кроме того, проблемы могут возникнуть со стороны заказчика при его нежелании обновлять продукт по тем или иным причинам, оставаясь при этом на старой версии.

Одним из таких решений является система *S-Market.* Она представляет собой систему управления торговым предприятием. Данное решение предоставляет полнофункциональную систему управления товародвижением, а также автоматизацию учета товаров в магазине, контроля прихода товаров и их остатков, анализа эффективности продаж. Стартовая страница сайта системы *S-Market* представлена на рисунке 1.5.



Рисунок 1.5 – Стартовая страница сайта системы *S-Market*

Система состоит из набора функциональных модулей. Наиболее интересным является модуль складского учета. Он предоставляет автоматизацию в виде информации о всех поступающих и уходящих на полки товарах, загружаемых в базу данных администратором системы. Кроме программного обеспечения *S-Market* предоставляет поддержку аппаратного комплекса, автоматизирующего учет продаваемых товаров и некоторых операций с ними.

*S-Market* также включает в себя аналитический модуль, который предоставляет возможность расчета скорости продаж, просмотра товародвижения по карточкам товаров, по клиентам и по поставщикам, анализ продаж и расчет себестоимости товаров и ежедневных остатков.

## Облачные решения

В отличие от программных решений, облачные решения предоставляют больше возможностей как для заказчика, так и для разработчика данного продукта. Существенными достоинствами данного решения являются такие факторы, как неограниченный объем дискового пространства, оперативной памяти и количества процессоров, доступ к продукту через веб-браузер, отсутствие зависимости от дорогого дополнительного оборудования, отсутствие нужды на развертывание инфраструктуры. Также, компании-заказчику не требуется выделять дополнительные средства на обучение персонала использованию продукта, а сами облачные системы обслуживаются в основном высококвалифицированными специалистами, что оказывает весьма положительное влияние на качество предоставляемых услуг.

Разумеется, облачные решения не лишены изъянов. Наиболее остро стоят вопросы фильтрации данных и безопасности их передачи. Заказчик не всегда готов доверить стороннему провайдеру все свои данные, из-за чего приходится тщательно отбирать, какие данные можно передавать и хранить на сервере. Также, от заказчика будет требоваться постоянный доступ в интернет. Еще одним недостатком является возможность потерять данные в результате произошедшего сбоя у поставщика услуг.

Примером облачного решения является система автоматизации розничной торговли *CloudShop*. В качестве особенностей данной системы можно выделить ведение автоматизированного учета продукции. Все данные по поступлению и реализации товара поступают в базу данных, что в значительной степени упрощает все организационные процессы, необходимые для оперативной работы торговой точки. Страница системы *CloudShop* представлена на рисунке 1.6.



Рисунок 1.6 – Страница системы *CloudShop*

Алгоритм действий системы *CloudShop*:

* Для каждой товарной позиции создается карточка товара, которая содержит все данные по поступлению и реализации данного товара;
* Реализация продукта производится путем выбора путем выбора карточки конкретного товара через терминал и отражения сведений об объеме покупки. Если позиций в корзине покупателя несколько, данные с нескольких карточек оформляются одним чеком;
* Как только продавец получил деньги от покупателя, покупка регистрируется, пробивается чек, и все данные по ней фиксируются в базе данных;
* Поскольку продукты имеют ограниченный срок годности, система оповещает о наличии товаров, срок реализации которых подходит к концу, или уже истек. В первом случае, продукты необходимо продать со скидкой, во втором – списать;

Также система позволяет вести контроль остатков продуктов на складе, а, следовательно, своевременно пополнять запасы и исключать позиции, на которые низок спрос [4].

## Аппаратные решения

Аппаратное решение практически всегда связанно с программным обеспечением. Сама по себе аппаратура может нести мало смысла без какой-либо обработки или анализа данных. В этом плане, техническая составляющая супермаркета может работать в связке как с обычным программным решением, так и с облачным. Однако, для учета количества товаров и их правильного перемещения между супермаркетом и его складом, зачастую бывают необходимы аппаратные решения.

Преимуществ от внедрения аппаратных средств довольно много. Прежде всего, это позволяет автоматизировать многие процессы при работе с товарами и их количеством. Кроме того, техническая составляющая дает возможность легче и быстрее вносить и редактировать данные о том или ином товаре, а также данные о количестве проданных товаров, что способствует правильному подсчету аналитических данных.

К сожалению, во внедрении аппаратной составляющей также присутствуют и некоторые недостатки. Прежде всего, препятствием для внедрения техники в работу супермаркета может являться его дороговизна. Аппаратные решения стоят немалых денег, и если внедрять их не на этапе начала работы супермаркета, придется просчитать целесообразность производимых на это затрат. К недостаткам также можно отнести сложность работы с аппаратной составляющей, то есть для правильного использования аппаратуры персоналу будет необходимо пройти некоторое обучение. Также, аппаратное решение по своей сути предполагает наличие какого-либо программного обеспечения, поэтому супермаркету в любом случае придется внедрять дополнительно и программное решение, что также может предполагать некоторые затраты.

Типичными аппаратными решениями для ведения учета товаров супермаркета являются терминалы сбора данных и сканеры штрих-кодов.

Терминал сбора данных – это портативный компьютер, необходимый для оперативного сбора, обрабатывания и передачи данных о товарах и изделиях. Основное предназначение терминала сбора данных – сбор данных о товарах и грузах и их последующая передача в систему учета, уменьшение временных затрат на выполнение многих рутинных операций, а также снижение количества ошибок учета. Внешний вид терминала сбора данных представлен на рисунке 1.7.



Рисунок 1.7 – Терминал сбора данных

Одним из поставщиков данного решения является компания ШТРИХ-М СПб. Терминалы для сбора данных, представленные в компании, обладают широким набором функций и максимально просты в применении. Использование таких приборов дает возможность обеспечить дополнительные удобства при обслуживании покупателей и соответственно существенно увеличить объем продаж [5].

Сканер штрих кода — это устройство, которое обеспечивает сканирование штрих-кода, перевод его графических элементов в цифровую последовательность, декодирование данных, проверку качества считывания и передачу полученной информации в компьютер, кассовый терминал [6]. Таким образом, данное устройство позволяет обеспечить обработку информации о каждом считываемом товаре для его последующего учета.

## Анализ существующих систем и инструментов

В рамках анализа существующих аналогов и прототипов разрабатываемого программного комплекса были изучены имеющиеся на рынке системы схожего предназначения. Многие из них специализируются такой функции, как построение оптимального плана выкладки товаров на полке для повышения продаж и увеличения прибыли. Кроме того, они предоставляют некоторый анализ уровня продаж. Однако, не удалось выявить систему, которая настраивала бы свои параметры в процессе работы и выдавала информацию по текущему состоянию продуктов и товаров на полках, а также рекомендации к действиям в определенный момент времени. Относительно данных факторов, можно сделать вывод, что разрабатываемая автоматизированная система предоставляет уникальные возможности в своей сфере.

## ABM Shelf

Данная система представляет собой облачное решение, разработанное компанией *ABM Cloud*. *ABM Shelf* предоставляет возможность составлять планограммы, моделировать залы и схемы выкладки в зависимости от разнообразных факторов. Кроме того, предоставляется возможность анализировать продажи магазина, в зависимости от чего выдаются статистические данные.

Система *ABM Shelf* предоставляет такие возможности, как конструктор оборудования, работу с базой данных магазинов, аналитику продаж торгового зала, автоматизацию управления выкладкой и управления торговым оборудованием, разработку планограммы выкладки товара, удобную визуальную аналитику по данным продаж и выкладки с возможностью экспорта в *Excel* [7]. Стартовая страница сайта системы *ABM Shelf* представлена на рисунке 1.8.

Система предоставляет много функционала по управлению полочным пространством. Данные о расположении товаров предоставляются в виде изображений торгового зала и каждого стеллажа по отдельности. Расположение товаров просчитывается автоматическими алгоритмами, благодаря чему выдаются оптимальные варианты расстановки товаров.

Также система предоставляет управление торговым пространством розничной сети с помощью анализа данных по мерчендайзингу товара, который предполагает анализ таких параметров, как текущий остаток товара, выкладка товара и оптимальное количество товара.



Рисунок 1.8 – Стартовая страница сайта системы *ABM Shelf*

К плюсам данной системы можно отнести:

* составление планограмм;
* моделирование торговых залов;
* аналитика продаж.

К минусам можно отнести:

* не всегда интуитивно понятный интерфейс;
* отсутствие взаимодействия со складом магазина.

## RS.ShelfSpace

*Retail Suite Shelfspace* — это система управления полочным пространством магазина. Она позволяет автоматизировать и настроить эффективное управление торговым пространством розничной сети. Реализован сценарный подход, который позволяет одновременно управлять несколькими задачами, запущенными параллельно. Торговое пространство розничной сети должно быть эффективным и управляемым. Для того, чтобы этого добиться, *RS.ShelfSpace* позволяет контролировать целый набор показателей визуального мерчандайзинга, отражающих качество использования торговых площадей [8].

Данная система обладает рядом преимуществ и недостатков. Прежде всего, она, возможно, могла бы предоставить больший функционал, однако по исходным данным видно, что система реализует лишь несколько общих вариантов автоматизации, без привнесения новизны в уже существующие методы. Так, например, данная система может составлять планограммы и возможные варианты расположения товаров только по заранее настроенным правилам, то есть это позволяет выставить определенный набор правил перед началом работы, относительно которого система выдаст возможные варианты без последующей возможности перенастройки системы в процессе работы. Кроме того, так как данная система анализирует данные только в конце рабочего дня, она не может сигнализировать об отсутствии определенного товара на полке, поэтому в данном случае остается человеческий фактор отслеживания заполненности полок.

В дополнение можно сказать, что данная система имеет довольно громоздкий интерфейс, предоставляющий информацию о стеллажах в виде списка типа «дерево». Данное решение может сказаться на интуитивном понимании интерфейса в случае разрастания списка. Пример конфигурации магазина представлен на рисунке 1.9. Стартовая страница сайта системы *RS.ShelfSpace* представлена на рисунке 1.10.



Рисунок 1.9 – Пример конфигурации магазина



Рисунок 1.10 – Стартовая страница сайта системы *RS.ShelfSpace*

Преимущества системы *RS.ShelfSpace*:

* автоматизация мерчендайзинга;
* создание планограмм;
* анализ продаж.

Недостатки данной системы:

* анализ данных происходит только в конце рабочего дня;
* громоздкий интерфейс.

## PlanoManager

Инструмент *PlanoManager* предназначен для создания планограмм и анализа их эффективности. Данное решение позволяет визуализировать торговое оборудование и размещение продукции, а также определить необходимый уровень товарного запаса для оценки и оптимизации выкладки. *PlanoManager* автоматически обновляет базу данных продуктов и автоматически строит планограммы на основе поступивших продуктов.

Данный инструмент поддерживает не так много функций, как предыдущие системы, однако он делает себя максимально эффективным в сфере планограмм. Кроме того, *PlanoManager* не предоставляет расширенной автоматизации на основе анализа продаж. Стартовая страница сайта инструмента *PlanoManager* представлена на рисунке 1.11.



Рисунок 1.11 – Стартовая страница сайта инструмента *PlanoManager*

К преимуществам данного инструмента можно отнести:

* приятный, простой и интуитивно понятный интерфейс;
* расширенный функционал для построения планограмм;
* автоматическое обновление базы данных продуктов (на основе данных, предоставляемых пользователем);
* визуализация планограмм.

К недостаткам можно отнести:

* отсутствие расширенного анализа данных;
* отсутствие автоматизации контроля полочного пространства.

## Постановка задачи

Разработать автоматизированную систему управления полочным пространством супермаркета, которая позволит автоматизировать ряд задач по работе с товарами, сократит расходы на персонал и улучшит качество обслуживания клиентов.

Система должна обладать возможностью визуализации текущего состояния супермаркета, анализировать поступающие данные и настраивать параметры своей работы в соответствии с полученными после обработки данными.

Система должна иметь базу данных для хранения и обработки поступающих данных о товарах и их перемещению по ключевым точкам супермаркета, а также для хранения и предоставления данных о статистике работы самой системы.

Необходимо, чтобы данная система была доступна для персонала супермаркета и обладала возможностью разграничения прав доступа к отдельным ее частям: управляющий персонал должны иметь доступ ко всем компонентам системы, авторизованные пользователи и рабочий персонал – к страницам «Состояние склада» и «Состояние магазина», неавторизованные пользователи – только к страницам «О системе» и «Авторизация». Доступ к системе должен предоставляться через веб-интерфейс.

1. ПРОЕКТИРОВАНИЕ СТРУКТУРЫ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ УПРАВЛЕНИЯ ПОЛОЧНЫМ ПРОСТРАНСТВОМ СУПЕРМАРКЕТА

## Структура разрабатываемой автоматизированной системы

Разрабатываемая автоматизированная система должна быть доступна пользователям в сети интернет. Кроме того, системе необходимо наличие базы данных. Исходя из этих фактов, для реализации системы была выбрала модель взаимодействия клиент-сервер.

Концепция клиент-сервер подразумевает взаимодействие двух сторон: клиента и сервера. В качестве клиента может выступать заказчик той или иной услуги, а в качестве сервера – поставщик услуг. Клиентская часть решает задачи отображения контента, серверная же часть, в свою очередь, содержит бизнес логику, основные алгоритмы и берет на себя большую часть вычислительных нагрузок. Кроме того, серверная часть осуществляет связь с базой данных приложения. Типичным клиентом является браузер, в качестве сервера может выступать как удаленный HTTP сервер, так и локальный веб-сервер.

К преимуществам клиент-серверной архитектуры можно отнести:

* программный код клиентского приложения и серверразделен;
* минимальные системные требования к компьютерам пользователей, т.к. вся аналитическая нагрузка вынесена на сервер.

Одним из недостатков данной архитектуры является то, что недоступность сервера по любой причине парализует работу всего программного комплекса.

Пользователь взаимодействует с клиентской частью приложения, которая при необходимости формирует запросы и обращается на сервер за данными, после чего принимает ответ и отображает данные.

Серверная часть, при поступлении на нее запроса, совершает обработку введенных данных, в большинстве случаев обращается в базу данных за данными, из которых, после анализа, формирует ответ и отправляет его на клиентскую часть.

Архитектура клиент-сервер не делит машины на только клиент или только сервер, а скорее позволяет распределить нагрузку и разделить функционал между клиентской частью и серверной. Архитектура приложения представлена на рисунке 2.1.



Рисунок 2.1 – Архитектура клиент-сервеного приложения

## Проектирование автоматизированной системы с использованием инструментальных средств

## Проектирование автоматизированной системы с использованием языка UML

*UML (Unified Modeling Language)* – это унифицированный язык объектно-ориентированного моделирования. Он служит для наглядного представления функциональности разрабатываемого программного продукта.

Для построения наиболее общей и абстрактной концептуальной модели системы используется диаграмма вариантов использования. Она описывает взаимодействие объектов автоматизированной системы между собой в процессе использования ее по назначению. Диаграмма вариантов использования автоматизированной системы представлена на рисунке 2.2.

В проектируемой автоматизированной системе можно выделить пятерых актеров: неавторизированный пользователь, персонал, менеджер, системный администратор, программное и аппаратное обеспечение.

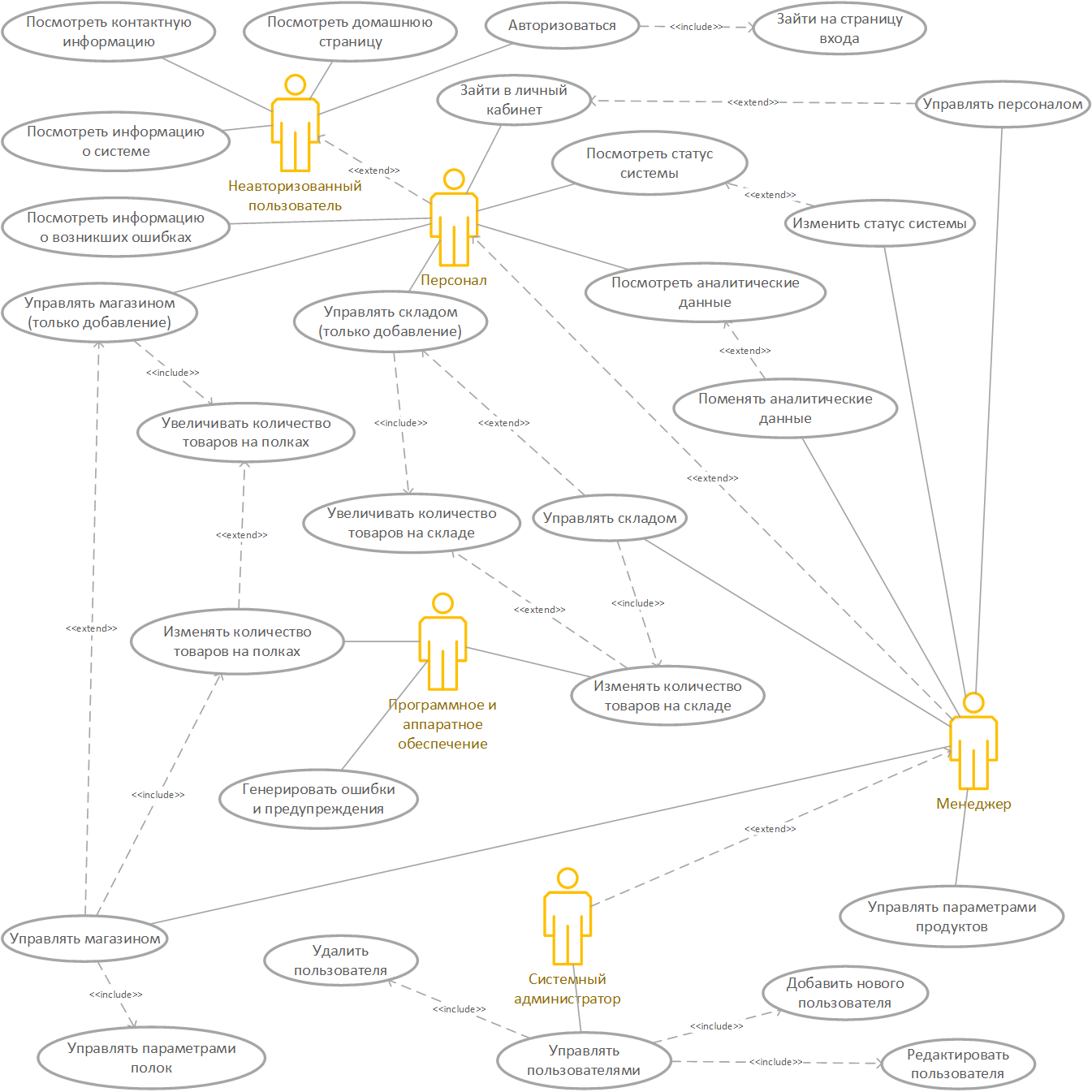


Рисунок 2.2 – Диаграмма вариантов использования автоматизированной системы

У каждого из актеров имеются свои варианты использования, которые также называются прецедентами. Прецедент – это типичное взаимодействие пользователя с системой, которое при этом:

* описывает видимую пользователем функцию;
* может представлять различные уровни детализации;
* обеспечивает достижение конкретной цели, важной для пользователя.

Рассмотрим варианты использования каждого актера. Возможности автоматизированной системы, доступные неавторизованному пользователю:

* авторизация, которая включает возможность перехода на страницу авторизации;
* просмотр домашней страницы;
* просмотр контактной информации владельца системы;
* просмотр общей информации о системе.

Персонал имеет все возможности неавторизованных пользователей. Также ему доступны такие функции, как:

* вход в личный кабинет для просмотра индивидуальной рабочей информации;
* просмотр текущего статуса системы;
* просмотр аналитических данных системы;
* ограниченное управление складом, которое подразумевает увеличение количества товаров на складе;
* ограниченное управление магазином, которое подразумевает увеличение товаров на полках;
* просмотр информации о возникших ошибках и предупреждениях.

Менеджер расширяет функционал персонала с добавлением таких возможностей, как:

* управление персоналом в рамках системы;
* изменение статуса системы;
* изменение аналитических данных и параметров системы;
* полное управление складом;
* полное управление магазином;
* управление продуктами, с которыми взаимодействует система.

В свою очередь, системный администратор обладает всеми перечесленными возможностями, включая управление пользователями системы. То есть именно системный администратор имеет возможность изменять параметры пользователей, а также добавлять и удалять их.

Кроме того, система подразумевает наличие сущности, не относящейся к живым аналогам, но обладающей возможностью субъективного взаимодействия с некоторыми частями системы и являющейся синтезом программного и аппаратного обеспечения. Данный актер обладает такими возможностями, как изменение количества товаров на полках и на складе, а также генерация ошибок и предупреждений, возникающих в процессе работы системы.

## Функциональная структура автоматизированной системы

Для формализации и описания бизнес-процессов используется методология функционального моделирования *IDEF*0. В *IDEF*0 рассматривается функциональная структура объекта, т.е. выполняемые обьектами действия и логические отношения между ними, а не их временная последовательность.

Функциональная модель *IDEF*0представляет собой набор блоков, каждый из которых имеет входы и выходы, управление и механизмы. Наиболее важная функция расположена в верхнем левом углу. Соединяются функции между собой при помощи стрелок и описаний функциональных блоков.

Контекстная диаграмма разрабатываемой автоматизированной системы изображена на рисунке 2.3.



Рисунок 2.3 – Контекстная диаграмма автоматизированной системы управления полочным пространством супермаркета

На вход системы подаются данные о количестве товаров на полках и на складе, а также данные о продажах, которые были произведены в определенный промежуток времени.

Правилами системы являются нормы количества товаров на полке, стандарты выкладки товаров и инструкции по обработке сигналов. Первые два правила дают описание регулирования товаров в зависимости от его избытка, недостатка и расположения. Третье правило задает порядок действий при возникновении определенных сигналов в процессе работы системы.

Механизмами управления являются аппаратное обеспечение, программное обеспечение и персонал супермаркета. Аппаратное обеспечение позволяет производить учет товаров в супермаркете при продаже и поступлении. Программное обеспечение выполняет хранение данных о магазине, производит подсчет проданных и поступивших товаров, а также производит аналитические вычисления. Персонал выполняет контролирующую функцию, то есть следит за корректным выполнением задач, возложенных на систему, а также заполняет полки.

Результатом работы системы являются заполненные товарами полки и полученные статистические данные. Полки будут считаться заполненными в том случае, когда соблюдены все правила, заданные при работе системы, а также при отсутствии ошибочных сигналов. Статистические данные представляют собой общие итоговые результаты работы системы, используемые для дальнейшего анализа.

Чтобы более подробно описать процессы, происходящие внутри автоматизированной системы, используется диаграмма декомпозиции, представленная на рисунке 2.4.

На диаграмме изображены 9 работ, которые будут выполняться в процессе функционирования системы.

Перед началом работы программное обеспечение вычисляет оптимальное количество товаров на полках. Для этого необходимы данные о текущем количестве, а также о предыдущих продажах. Оптимальное количество вычисляется с ориентировкой на нормы допустимого количества товаров.

Вычисленные данные проходят на следующий этап, на котором аппаратное обеспечение во взаимодействии с программным подает сигнал либо об избыточном количестве товаров на полке, либо о недостаточном. Данные сигналы записываются в раздел данных, из которого будет формироваться статистический результат работы.

****

Рисунок 2.4 – Диаграмма декомпозиции автоматизированной системы управления полочным пространством супермаркета

При сигнале о недостаточном количестве товара на полке персоналом выполняется проверка наличия товара на складе. В случае отсутствия товара, подается сигнал, записывающийся в раздел статистики. При наличии товара также подается сигнал, после чего персонал забирает определенное количество со склада.

Полученный на складе товар размещается в соответствии со стандартами выкладки товаров на полке. Измененное количество товара записывается в окончательную статистику.

При сигнале об избыточном количестве товара на полке, персонал забирает товар с полки до тех пор, пока его количество не станет оптимальным. Товар с полки возвращается на склад. Измененное количество товара на складе передается в статистические данные.

По окончанию выполнения данного процесса формируется статистический результат работы системы. На выходе получаются сформированные на основе работы системы статистические данные.

Исходя из того, что в процессе работы автоматизированной системы подразумевается взаимодействие с информацией, имеет смысл построение диаграммы на основе *DFD* методологии для этапа работы с базой данных. Такая диаграмма представлена на рисунке 2.5.



Рисунок 2.5 – Диаграмма декомпозиции процесса работы с базой данных

На диаграмме декомпозиции процесса работы с базой данных изображены потоки данных, поступающих на обработку, на ввод и в качестве результатов. Все данные поступают изначально от персонала. После этого различные модули достают необходимую информацию и обрабатывают ее в соответствии с определенными правилами. Результирующими данными будут являться статистические данные, которые сохраняются в базу данных и выводятся на экран персоналу. Кроме того, в процессе работы системы персоналу будут приходить уведомления о тех или иных неполадках.

## Информационное обеспечение автоматизированной системы

## Входные данные

Разрабатываемая автоматизированная не предполагает наличие системы регистрации новых пользователей, так как количество пользователей ограничено персоналом супермаркета. Для создания нового пользователя используются возможности панели администратора, взаимодействовать с которой может системный администратор.

Входные данные для нового пользователя:

* логин пользователя;
* пароль пользователя;
* электронная почта;
* группа, к которой пользователь принадлежит;
* дата и время последнего входа.

Входные данные автоматизированной системы представляют собой некоторые качественные и количественные характеристики супермаркета и продуктов, необходимые для работы системы.

Входные данные по продуктам:

* код продукта;
* размеры продукта;
* название продукта;
* цена за единицу продажи;
* срок годности;
* вес единицы продажи.

Входные данные по параметрам полок:

* относящийся к полке продукт;
* грузоподъемность полки;
* размеры полочного пространства;
* количество единиц продажи;
* название полки;
* название секции, к которой относится полка.

Дополнительные входные данные относятся к ячейкам склада, такие как продукт, относящийся к ячейке, дата поставки, количество продукта в ячейке, и акционным товарам, такие как продукт, на который распространяется акция, полка, к которой отностится данная акция, сроки проведения акции и цена единицы продукта по акции.

Источником входных данных являются персонал и менеджер супермаркета. Некоторые входные данные, такие как даты и необязательные поля, генерируются автоматически. Периодичность поступления данных отсутствует, так как данные могут быть занесены в любое время любым пользователем системы.

## Проектирование базы данных

Разрабатываемая автоматизированная система взаимодействует с базой данных. База данных представляет из себя совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняется в соответствии с правилами средств моделирования данных. В базе данных разрабатываемой автоматизированной системы содержится 8 сущностей: пользователи, продукт, магазин, акция, продажи, склад, сбои, статистика.

*Django* использует тезнологию объектно-реляционного отображения для упрощения взаимодействия базы данных и программного кода. Данная технология позволяет автоматизировать некоторые процессы создания сущностей базы данных и их экземпляров, представляя возможные поля сущностей в виде связанной модели. Такой подход упрощает создание связей между сущностями, а также позволяет упростить изменение полей и связей сущности.

Сущность «*User*» содержит информацию о каждом зарегистрированном пользователе. Ее структура приведена в таблице 2.1.

Таблица 2.1 – Структура сущности *«User»*

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| *Id* | *AutoField* | Идентификатор пользователя |
| *Email* | *CharField* | Электронная почта пользователя |
| *Group* | *CharField* | Группа пользователя |
| *Last\_access* | *DateTimeField* | Дата и время последнего входа |
| *Login* | *CharField* | Логин пользователя |
| *Password* | *CharField* | Пароль пользователя |

Сущность «*Failure*» содержит информацию о возникших в процессе работы системы ошибках, возможных причинах и способах устранения. Ошибки в таблице помечаются исправленными, если поле *is\_solved* содержит значение *True*. Кроме того, каждой ошибке может присваиваться пользователь, которому поручено ее устранить. За информацию о пользователе, которому выдано задание на устранение ошибки, отвечает поле *assignee*. Структура таблицы «*Failure*» приведена в таблице 2.2.

Сущность «*Product*» содержит информацию о продуктах, когда-либо заносимых в систему. Данная сущность предоставляет о параметрах продуктов, их текущей цене, сроке годности и других характеристиках. Ее структура приведена в таблице 2.3.

Таблица 2.2 – Структура сущности *«Failure»*

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| *Id* | *AutoField* | Идентификатор неполадки |
| *Assignee* | *ForeignKey (id)* | Идентификатор пользователя, которому назначено решение неполадки |
| *Is\_solved* | *BooleanField* | Статус решения ошибки |
| *Possible\_cause* | *TextField* | Описание возможного источника |
| *Possible\_solution* | *TextField* | Описание возможного решения |
| *Severity* | *IntegerField* | Серьезность ошибки |
| *Stack\_trace* | *TextField* | Трассировка программного кода, приведшего к ошибке |
| *Text* | *TextField* | Описание ошибки |
| *Timestamp* | *DateTimeField* | Дата и время возникновения ошибки |

Таблица 2.3 – Структура сущности *«Product»*

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| *Id* | *AutoField* | Идентификатор продукта |
| *Digit\_code* | *CharField* | Цифровой код продукта |
| *Height* | *FloatField* | Высота продукта в миллиметрах |
| *Length* | *FloatField* | Длина продукта в миллиметрах |
| *Width* | *FloatField* | Ширина продукта в миллиметрах |
| *Name* | *CharField* | Название продукта |
| *Price* | *FloatField* | Цена продукта |
| *Shelf\_life* | *DateTimeField* | Срок годности продукта |
| *Stackable* | *BooleanField* | Возможность складывать продукт по высоте |
| *Weight* | *FloatField* | Вес продукта |

Сущность «*Storage*» содержит информацию о продуктах, хранящихся на складе. Данная сущность связана с таблицей «*Product*» и предоставляет информацию о дате и времени последней поставки продуктов на склад, а также текущем количестве на складе. Ее структура представлена в таблице 2.4.

Сущность «*Sales*» связана с сущностью «*Product*»и содержит информацию о продажах продуктов, такую как цена на момент продажи, дата и время продажи, количество проданных продуктов. Структура данной сущности представлена в таблице 2.5.

Таблица 2.4 – Структура сущности *«Storage»*

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| *Id* | *AutoField* | Идентификатор ячейки склада |
| *Product* | *ForeignKey (id)* | Идентификатор продукта |
| *DeliveryTimestamp* | *DateTimeField* | Дата и время последней поставки |
| *Product\_count* | *IntegerField* | Текущее количество продуктов на складе |

Таблица 2.5 – Структура сущности *«Sales»*

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| *Id* | *AutoField* | Идентификатор продажи |
| *Product* | *ForeignKey (id)* | Идентификатор продукта |
| *Sold\_count* | *IntegerField* | Количество порданных продуктов |
| *Sold\_date* | *DateTimeField* | Дата и время продажи |
| *Sold\_price* | *FloatField* | Цена продукта на момент продажи |

Сущность «*Store*» содержит данные о каждой полке магазина. Данная сущность связана с таблицами «*Product*» и «*Stock*» и предоставляет информацию о названии полки, названии секции, к которой относится полка, параметрах полки, продукте, который хранится на полке в данный момент, его количестве и последней загрузке, а также о максимально допустимом весе, который выдерживает полка. Структура данной таблицы представлена на рисунке 2.6.

Сущность «*Stock*» хранит данные об акциях, предусмотренных в супермаркете. Эта сущность также связана с таблицами «*Product*» и «*Store*» и предоставляет информацию о продукте, полке, к которой относится акция, дате и времени начала и конца акции, а также о цене продукта во время проведения акции. Структура данной сущности представлена в таблице 2.7.

Сущность «*Statistics*» содержит аналитические данные на каждый день работы супермаркета. Она предоставляет информацию о количестве произошедших неполадок, доходе, количестве проданных продуктов и о том, проводились ли в данный день акции. Структура данной сущности приведена в таблице 2.8.

Таблица 2.6 – Структура сущности *«Store»*

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| *Id* | *AutoField* | Идентификатор полки |
| *Product* | *ForeignKey (id)* | Идентификатор продукта |
| *Carrying\_capacity* | *FloatField* | Максимально допустимый вес продуктов на полке |
| *Height* | *FloatField* | Высота полки в миллиметрах |
| *Length* | *FloatField* | Длина полки в миллиметрах |
| *Width* | *FloatField* | Ширина полки в миллиметрах |
| *Last\_charge* | *DateTimeField* | Дата и время последней загрузки полки |
| *Product\_count* | *IntegerField* | Текущее количество продуктов на полке |
| *Section\_name* | *CharField* | Название секции, в которой расположена полка |
| *Shelf\_name* | *CharField* | Название полки |

Таблица 2.7 – Структура сущности *«Sales»*

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| *Id* | *AutoField* | Идентификатор акции |
| *Product* | *ForeignKey (id)* | Идентификатор продукта |
| *Shelf* | *ForeignKey (id)* | Идентификатор полки |
| *Start\_timestamp* | *DateTimeField* | Дата и время начала проведения акции |
| *End\_timestamp* | *DateTimeField* | Дата и время конца проведения акции |
| *Stock\_price* | *FloatField* | Цена продукта во время проведения акции |

Таблица 2.8 – Структура сущности *«Statistics»*

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Описание |
| *Id* | *AutoField* | Идентификатор записи |
| *Day* | *DateField* | Дата, на которую составлена статистика |
| *Failures\_count* | *IntegerField* | Количество произошедших неполадок |
| *Revenue* | *DecimalField* | Доход в данный день |

Продолжение таблицы 2.8

|  |  |  |
| --- | --- | --- |
| *Is\_stock\_day* | *BooleanField* | Индикатор проведения акций в данный день |
| *Sold\_count* | *IntegerField* | Количество проданных продуктов в данный день |

На рисунке 2.6 представлена структурная схема используемой базы данных.

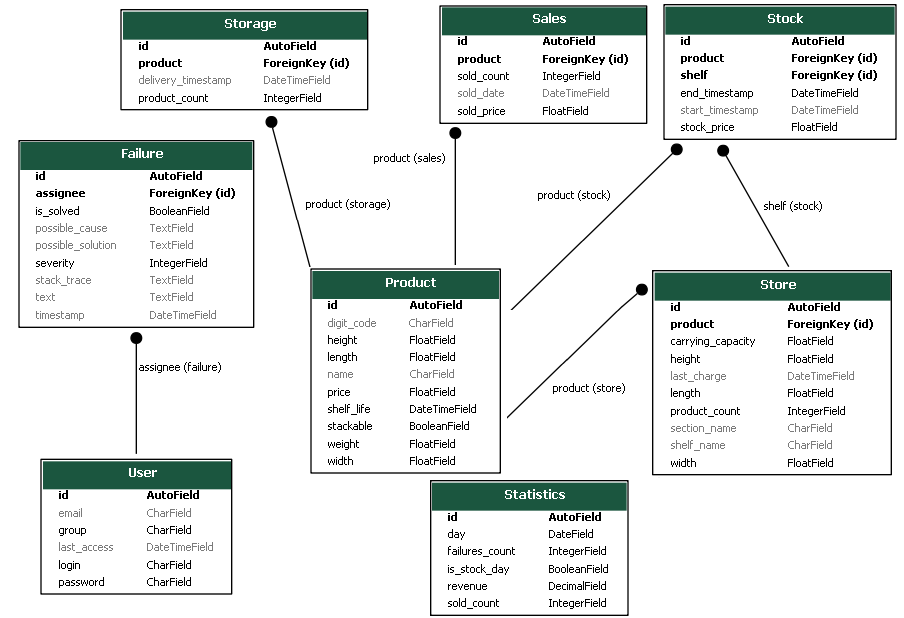


Рисунок 2.6 – Структурна схема базы данных

## Выходные данные

Выходными данными программного комплекса являются два типа информации: значение нормального артериального давления для неавторизированного пользователя, рекомендации и история для авторизованного пользователя.

Для неавторизованного пользователя выходная информация будет состоять из значения нормального артериального давления

Для авторизованного пользователя выходные данные будут состоять из следующего:

* История значений артериального давления (графическое отображение);
* История значений артериального давления (числовое отображение);
* Список рекомендаций и диагнозов пользователю.

## Алгоритмическое обеспечение

Исходя из того, что одной из основных целей работы разрабатываемого программного комплекса является анализ входных данных, имеет смысл описать алгоритм, который используется для анализа и формирования выходных данных.

Ниже представлены блок-схемы функционирования программного комплекса для всех видов пользователей.

Блок-схема функционирования программного комплекса для неавторизованного пользователя представлена на рисунке 2.7.

Если говорить о неавторизованном пользователе, то после ввода данных в форму и ее подтверждения на сервер отправляется запрос с введенными данными, сервер на основе полученных данных проверяет пол, затем для выбранного пола по введенному возрасту определяет нормативное значение и отправляет его в ответе клиенту, который в свою очередь отображает полученную информацию.

Первый этап для авторизованного пользователя аналогичен, это отправка введенных значений на сервер. Сервер извлекает данные из запроса и записывает их в базу данных, используя как *id* записи токен пользователя. Затем происходит выборка из базы данных всех значений с текущим токеном, таким образом мы получаем историю значений конкретного пользователя. После этого для каждой записи истории мы находим нормативное значение артериального давления на основе данных этой записи и добавляем это значение в эту запись. Затем для каждой записи истории мы запускаем модуль анализа, который пройдется по всем данным текущей записи и на основе их значений добавит в эту запись список рекомендаций и возможных причин отклонения значений артериального давления от нормы, после чего сервер отправит эти данные на клиент. Клиент, получив эти данные, визуализирует их, построив график отклонения значений артериального давления от нормы, так и просто в текстовом формате. Кроме того он отобразит список выявленных рекомендаций, причин и потенциальных диагнозов.

Блок-схема функционирования программного комплекса для авторизованного пользователя представлена на рисунках 2.8 и 2.9.



Рисунок 2.7 – Блок-схема функционирования программного комплекса для неавторизованного пользователя



Рисунок 2.8 – Блок-схема функционирования программного комплекса для авторизованного пользователя



Рисунок 2.9 – Блок-схема функционирования программного комплекса для авторизованного пользователя

## Техническое и системное программное обеспечение

Так как программный комплекс имеет архитектуру клиент-сервер, а клиентская часть запускается в браузере и написана с использованием современных решений в сфере веб-разработки, таких как *React* и *Redux*, минимальными системными требованиями со стороны клиента можно назвать минимальные системные требования браузера, поддерживающего эти технологии.

Серверная часть приложения также не производит сколь-либо тяжелых вычислительных операций, поэтому изначально не требуется больших вычислительных мощностей для ее функционирования. Однако с учетом ожидаемого ежедневного роста базы данных со значениями, а соответственно ежедневного увеличения сложности выборки данных из базы с последующей обработкой и анализом каждой записи, потенциально необходимость в вычислительных мощностях может увеличиться.

Ниже приведены предположительные минимальные системные требования.

Системные требования для пользователя:

* процессор *Intel® Core™ i*3*–*2350*M* 2300 *МHz* или более мощный;
* ОЗУ емкостью от 256 Мб;
* обязательно необходим выход в Интернет;
* встроенная видеокарта 256 – 1024 Мб или больше;
* монитор желательно с разрешением 1024х768 или больше;
* клавиатура, мышь.

Системные требования для реализации сервера:

* процессор *Intel® Core™ i*5*–*3230*M* 2.60 *GHz* или более мощный;
* ОЗУ емкостью от 512 Мб;
* обязательно необходим выход в Интернет;
* видеокарта встроенная;
* клавиатура, мышь.

## Эргономическое обеспечение

Эргономика программного обеспечения – это степень, в которой продукт может быть использован пользователем для достижения поставленных целей эффективно и с удовольствием в заданном контексте использования.

Под эргономикой программного обеспечения в большинстве случаев понимают такие вещи, как:

* удобство интерфейса;
* доступность и логичность размещения элеметнов упраления;
* оптимальные цветовые решения.

Для решения вопросов эргономичности сквозь весь программный комплекса прослеживается одна структура размещения окон. Все формы, сменяющие друг друга, расположены на одном месте, в центре, куда и направлен взгляд пользователя. Кнопки навигации по страницам расположены сразу под формами и сменяют друг друга на одном и том же месте, что исключает необходимость отыскивать навигационные ссылки по углам сайта. Кроме того, такой выбор структуры сайта позволил сохранить десктопный и мобильный дизайны максимально схожими, что также облегчает использование сайта пользователям, использующим и мобильную, и настольную версию. Благодаря адаптивности реализована мобильная версия, что позволяет значительно увеличить удобство использования сайта с мобильных устройств.

Для оформления страниц выбран единый стиль материального дизайна, привычный современным пользователям, с узкими и элегантными, но легко различаемыми и читаемыми белыми шрифтами на темном фоне. В качестве шрифтов для шрифта заголовка был выбран *Dosis*, а для основного текста сквозь все страницы используется *Raleway*. Также шрифт *Dosis* был выбран для текста на кнопках. Для сообщений об ошибках выбран темно-красный цвет, привлекающий внимание, однако не давящий на глаза.

Фон для форм выбран мягкий темно синий, чтобы не резать глаза пользователю и улучшить читаемость белых надписей. С точки зрения восприятия глазом, было принято решение слегка зажелтить остальной фон, так как желтый свет гораздо легче воспринимается глазами человека.

Фон для графика был выбран прозрачным с еле видимой сеткой, и темными значениями, так как акцент должен идти не на сами значения, а на динамику их изменений и отклонений с течением времени. Сами же значения, для большей наглядности, продублированы в цифровом виде.

Ключевым аспектом эргономики является отсутствие каких-либо раздражителей либо вещей, которые могут вызвать недовольство либо дискомфорт у пользователя. Чтобы достичь этого, были убраны любые моргающие эффекты, которые могут отвлекать пользователя, отсутствует банерная реклама.

## Организационное обеспечение

Организационное обеспечение – это совокупность методов и средств, регламентирующих взаимодействие работников с техническими средствами и между собой в процессе разработки и эксплуатации программного комплекса.

В данном дипломном проекте к средствам, регламентирующим взаимодействие пользователей с ситемой будут являться права пользователя.

Предполагается наличие двух видов пользователей, имеющих различные права:

* авторизованные пользователи;
* неавторизованные пользователи.

Тип пользователя будет определяющим фактором при определении доступной ему функциональности.

Так неавторизованный пользователь будет иметь доступ только к начальному экрану с первичной формой анализа, а также функциям регистрации и авторизации.

Авторизованный же пользователь, помимо начального экрана с первичной формой анализа, будет иметь доступ к своему личному кабинету, где и распологается более интересная функциональность, а также истории своих введенных данных.

1. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ УПРАВЛЕНИЯ ПОЛОЧНЫМ ПРОСТРАНСТВОМ СУПЕРМАРКЕТА

## Обоснование выбора средств разработки

Процесс создания веб-сайта называется веб-разработкой. Он включает в себя верстку веб-страниц, веб-программирование, а также конфигурирование веб-сервера. Для каждой из перечисленных задач используются различные средства разработки.

Типичный веб-сайт представляет из себя текстовый файл в формате *HTML*, который может содержать ссылки на файлы в других форматах, а также ссылки на другие веб-сайты. По факту для саздания статичного веб-сайта достаточно только *HTML*, однако выглядят статические веб-сайты довольно скудно, к тому же они не обладают какой-либо приемлимой функциональностью, и разумеется не могут предоставить какой-либо возможности обратной связи с сервером.

Слегка оживить статические сайты, наполнить их какой-либо простой анимацией, расположить элементы а также украсить их внешне позволяет *CSS*. *CSS* – это формальный язык описания внешнего вида документа. Использоюется для задания цветов, изменения шрифтов, расположения элементов, создания простых анимаций.

Однако, чтобы сделать сайт хоть сколько-нибудь полезным и функциональным, придется воспользоваться одним из множества языков программирования.

Языки веб-программирования делятся на две группы: клиентские и серверные.

Клиентские языки программирования используются при *front-end* разработке, они обрабатываются на стороне клиента пользователя, т.е. программы на клиентском языке обрабатывает браузер. Отсюда следует недостаток: обработка скрипта зависит от браузера пользователя, и он в свою очередь имеет полномочия настроить свой браузер так, чтобы он игнорировал написанные программистом скрипты. При этом, если клиент использует старую версию браузера, то он может не поддерживать тот или иной язык или версию языка. С современными браузерами такие проблемы возникают редко, к тому же языки программирования обновляются сравнительно не часто и не кардинально, а лучшие из них давно известны.

Преимуществом же клиентского языка является то, что обработка скриптов на таком языке может выполняться без отправки документа на

сервер. Отсюда же вытекает и то ограничение, что с помощью клиентского языка программирования ничто не может быть записано на сервер, то есть, например, с его помощью нельзя сделать гостевую книгу, потому что тогда надо записывать сообщения в какой-либо файл на сервере.

Самым распространенным из клиентских языков является *JavaScript*, он так же, как и *HTML*, лежит в основе многих веб-технологий. На сегодняшний день этот язык является одним из самых популярных языков программирования, получил широчайшее распространение и его по умолчанию поддерживают практически все браузеры, операционные системы и мобильные устройства.

*Back-end* разработка подразумевает под собой создание серверной части приложения при использовании серверных языков программирования, которые соответственно работают на стороне сервера. Серверная часть приложения несет ответственность за взаимодействие с БД, архитектуру и программную логику.

К серверным языкам программирования можно отнести такие языки, как *PHP*, *JAVA*, *Perl*, *Python*, *Ruby*, а также любой .*NET* язык программирования. Все они имеют свои особенности и преимущества, однако в данном дипломном проекте будет использоваться *JavaScript* и на сервеной стороне. В этом на помощь приходит *Node.js*, который представляет из себя программную платформу, основанную на движке *v8*, превращающем *JavaScript* из узкоспециализированного языка в язык общего назначения, и выполняет роль веб-сервера. В качестве библиотеки будет использоваться *EXPRESS* – гибкая библиотека для *Node.js* приложений, предоставляющая большую функциональность для создания веб-приложений и *API*.

Среди систем управления базами данных, которые являются совокупность проммных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных, по модели данных выделяют следующие классификации: иерархические, сетевые, реляционные, объектно-ориентированные и объектно-реляционные.  
 Ярким представителем реляционных БД является *MySql*. Она подходит для огромного спектра задач, это проверенная и старая технология, которая используется крупными компниями вот уже 15 лет. Она обладает хорошей производительность, достойной расширяемостью, и проста в администрировании. Данные она хранит в виде таблиц, со строками и столбцами. Эти таблицы могут иметь связи между собой, что значительно упрощает работу с ними в некоторых случаях.

Если в *MySql* данные содержатся в таблицах, то в нереляционной MongoDB база данных состоит из коллекций, которые не имеют связей между собой. Это документоориентированная БД, в которой каждый документ можно представить как объект. Значительно упрощает работу с ней также формат данных, с которым она работает, а конкретно *JSON*.

Для создания веб-сайта были выбраны *HTML* и *CSS*, как не имеющие достойных аналогов средства написания веб-страниц. В качестве языка программирования для данного дипломного проекты был выбран язык *JavaScript*. Данный выбор был обусловлен огромной популярность *JavaScript*, его обширной поддержкой со стороны как ит-компаний, так и программистов, наличием необходимой функциональности как для написания *front-end* части, так и для *back-end* разработки. Кроме того, данный язык распростроняется на бесплатной основе и имеет открытый исходный код.

*JavaScript* – прототипный язык программирования. Он является реализацией языка *ECMAScript*(стандарт *ECMA*-262).

Обычно *JavaScript* используется в браузерах, как язык сценариев для приданий веб-сайтам интерактивности.

В связке с *Javascript* будут использоваться библиотеки *React* и *Redux*. *React* – это *JavaScript*-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов. *React* может использоваться для разработки одностраничных и мобильных приложений. Он позволяет разбивать страницу на мелкие части, представляющие из себя независимые компоненты. *React* использует *JSX* синтаксис, который является некоей интерпретацией *HTML* внутри *JavaScript*. В качестве библиотеки для разработки пользовательских интерфейсов *React* часто используется с другими библиотеками, такими как *Redux*.

В качестве серверного языка приложения был выбран все тот же *JavaScript*, который будет использоваться в сочетании с такими библиотеками как *Node.js* и *express*.

В качестве СУБД будет использоваться *MongoDB*. Такой выбор обусловлен опытом работы автора с данной системой. Кроме того, она отлично подходит для работы с JavaScript, имеет гибкий формат документов *JSON*, работа с которым автор находит гораздо более приятным, чем с таблицами, колонками и столбцами. Кроме того проблема производительности при написании крупных запросов в данном случае легко решается на стороне преложения, что также является хорошим решением.

## Структура программного продукта

Перед проектированием сложного программного продукта необходимо определиться с его структурой. Для этого нужно определить структурные компоненты программного продукта, а также связи между ними, а после этого составить структурную схему.

Структурной схемой называют схему отражающую состав и взаимодействие частей разрабатываемого программного обеспечения.

Структурная схема программного комплекса анализа артериального давления представлена на рисунке 3.1.



Рисунок 3.1 – Структурная схема программного комплекса

## Проектирование программной реализации комплекса

В качестве архитектурного подхода клиентской части программного комплекса была выбрана *Flux*-архитектура. *Flux*-архитектура – это архитектурный подход для построения пользовательского интерфейса веб-приложения, сочетающийся с реактивным программированием и построенный на однонаправленных потоках данных.

Основной отличительной особенностью *Flux* явлется односторонняя направленность передачи данных между компонентами *Flux*-архитектуры. Архитектура накладывает ограничения на поток данных, исключая возможность обновления состояния компонентов самими собой. Такой подход позволяет отслеживать причины изменений и возможных ошибок, значительно облегчая их обнаружение.

В реализуемом программном комплексе *Flux*-архитектура содержит четыре слоя, взаимодействующие между собой по порядку:

* *Actions* (действия);
* *Dispatcher* (диспетчер);
* *Stores* (хранилища);
* *Views* (представления);

Действия – это выражения событий. По факту пресдавляют из себя объект, содержащий тип действия и данные, с которыми в процессе этого действия необходимо будет манипулировать.

Диспетчер – это функция, которая содержит список зависимостей вызова определенных функций от типа действия и в зависимости от типа переданного действия определяет, какая функция будет вызван для манипуляции с переданными данными.

Хранилище – это место сосредоточения данных приложение, где хранится его состояние. Его обновление происходит строго на основе его предыдущего состояния и данных действия.

Представление – это компонент, главное предназначение которого в отражении данных пользователю.

Для *Flux*-архитектуры важно, чтобы данные с представлений попадали в хранилища только через действия.

ДОПИШИ !!!!!!!

## Описание программной реализации отдельных частей комплекса

## Модуль регистрации нового пользователя

Этот модуль служит для добавления в систему новых пользоветелей. После введения почты, повторения нового пароля и введения города, если пользователь желает получать уведомления, происходит запрос на сервер. На сервере происходит проверка на наличие пользователей с аналогичной почтой, после чего, при успешном прохождении проверки, сервер заносит эту запись в базу данных пользователей. В ответ же он возвращает уникальный идентификатор пользователя, который будет использоваться при последующем использовании приложения пользователем.

Форма регистрации новых пользователей представлена на рисунке 3.2.

На рисунке 3.3 представлена форма регистрации новых пользователей с сообщением о том, что пользователь с такой почтой уже имеется в базе.

По нажатии кнопки «*Submit*» происходит отправка данных на сервер:

*formSubmit(e) {*

*e.preventDefault();*

*let form = e.target;*

*if (form[1].value === form[2].value) {*

*let data = {*

*username: form[0].value,*

*password: form[1].value,*

*city: form[2].value,*

*};*

*fetch(constants.register,{*

*method: 'POST',*

*headers:{'content-type': 'application/json'},*

*body: JSON.stringify(data)*

*})*

*.then(function(resp) {*

*return resp.json();*

*})*

*.then((resp) => {*

*that.props.updateToken(resp.token);*

*that.props.history.push('/personalcab');*

*})*

*.catch((error) => {*

*this.setState({error: 'User with such mail is already registered'});*

*});*

*} else {*

*this.setState({error: 'entered passwords are different'});*

*}*

*}*

На сервере происходит следующая обработка:

*apiRouter.post('/register', (req, res) => {*

*usersModel.findOne({'username': req.body.username}, (error, user) => {*

*if (!user) {*

*usersModel.register(new usersModel({*

*username: req.body.username,*

*city: req.body.city*

*}), req.body.password, function(err, user){*

*if (err) {return res.send({ message: err });}*

*});};*

*});*

*});*

## Модуль авторизации пользователей

Модуль авторизации пользователей служит для предоставления зарегистрированным пользователям идентификационного токена, который необходим для доступа к личному кабинету пользователя.

Он представляет из себя форму с полями для ввода почты и пароля. Форма авторизации представлена на рисунке 3.4. При подтверждении формы происходит формирование запроса и отправка данных на сервер:

*formSubmit(e) {*

*e.preventDefault();*

*let that = this;*

*let form = e.target;*

*let data = {*

*username: form[0].value,*

*password: form[1].value*

*};*

*fetch(constants.login,{*

*headers:{'content-type': 'application/json'},*

*body: JSON.stringify(data)*

*})*

*.then(function(resp) {*

*return resp.json();*

*})*

*.then((resp) => {*

*that.props.updateToken(resp.token);*

*that.props.history.push('/personalcab');*

*});*

*}*

Сервер проверяет наличие соответствующего пользователя в базе данных, и в случае успеха возвращает его токен:

*apiRouter.post('/login', passport.authenticate('local'), function(req, res) {*

*usersModel.findOne({ 'username': req.body.username }, (error, user) => {*

*res.send({token: user.\_id});*

*});*

*});*

## Модуль определения нормативного артериального давления

Этот модуль предназначен для неавторизованных пользователей и представляет из себя форму с полями для ввода возраста и выбора пола. Форма определения нормативного давления представлена на рисунке 3.5. При потверждении формы происходит отправка данных на сервер:

*formSubmit(e) {*

*e.preventDefault();*

*let that = this;*

*let form = e.target;*

*let data = {*

*age: form[0].value,*

*gender: form[1].value*

*};*

*fetch(constants.calculatePressureURL,{*

*method: 'POST',*

*headers:{'content-type': 'application/json'},*

*body: JSON.stringify(data)*

*})*

*.then(function(resp) {*

*return resp.json();*

*})*

*.then(function(resp) {*

*that.setState({pressureValue: resp.normalValue});*

*})*

*}*

Сервер на основе поступивших данных определяет нормативное для такого сочетания данных значение и отправляет его на клиент:

*getNormalPressure: (req, res) => {*

*let data = {*

*age: req.body.age,*

*gender: req.body.gender,*

*systolicBloodPressure: req.body.systolicBloodPressure,*

*diastolicBloodPressure: req.body.diastolicBloodPressure*

*};*

*let bpOnGender = bpMocks[data.gender];*

*let normalValue;*

*for (key in bpOnGender) {*

*if (data.age > key) {*

*normalValue = bpOnGender[key];*

*}*

*}*

*data.normalValue = normalValue;*

*res.send(data);*

*},*

## Модуль анализа значений артериального давления

Этот модуль представляет из себя форму и представлен на рисунке 3.7. На вход в данный модуль поступает информация о пользователе. После подтверждения формы эта информация отправляется на сервер для дальнейшего анализа:

*formSubmit(e) {*

*e.preventDefault();*

*let that = this;*

*let form = e.target;*

*let value = form[2].value + '/' + form[3].value;*

*let data = {*

*token: this.state.token,*

*age: form[0].value,*

*gender: form[1].value,*

*value: value,*

*date: +new Date(),*

*height: form[4].value,*

*weight: form[5].value,*

*physicalActivity: form[6].value,*

*alcohol: form[7].value,*

*smoke: form[8].value*

*};*

*let xhr = new XMLHttpRequest();*

*xhr.open('PUT', constants.addvalue);*

*xhr.setRequestHeader('Content-Type', 'application/json');*

*xhr.onreadystatechange = () => {*

*if (xhr.readyState === 4) {*

*that.getUserHistory();*

*}*

*};*

*xhr.send(JSON.stringify(data));*

*}*

На сервере происходит сохранение полученного значения в базу данных:

*apiRouter.put('/addvalue', function(req, res, next) {*

*let data = new pressureModel({*

*token: req.body.token,*

*value: req.body.value,*

*age: req.body.age,*

*gender: req.body.gender,*

*date: req.body.date,*

*height: req.body.height,*

*weight: req.body.weight,*

*physicalActivity: req.body.physicalActivity,*

*alcohol: req.body.alcohol,*

*smoke: req.body.smoke*

*});*

*data.save();*

*res.send();*

*});*

Затем, после того как сервер сообщил, что данные сохранены, клиент делает запрос за всей историей пользователя:

*getUserHistory() {*

*let that = this;*

*fetch(constants.gethistory + '/' + this.state.token,{*

*method: 'GET',*

*headers:{'content-type': 'application/json'}*

*})*

*.then(function(resp) {*

*return resp.json();*

*})*

*.then(function(resp) {*

*that.setState({history: resp});*

*})*

*}*

Сервер делает выборку по идентификатору пользователя и добавляет к этим значениям значение нормативного артериального давления для каждой записи, после чего передает историю пользователя для анализа:

*apiRouter.get('/gethistory/:token', function(req, res, next) {*

*let token = req.params.token;*

*pressureModel.find({'token': token}, (error, items) => {*

*if (items) {*

*let array = [];*

Ниже происходит определение нормативного значения артериального давления в зависимости от пола и возраста пользователя:

*items.forEach(function(item) {*

*let bpOnGender = bpMocks[item.gender];*

*let normalValue;*

*let obj;*

*for (key in bpOnGender) {*

*if (item.age > key) {*

*normalValue = bpOnGender[key];*

*}*

*}*

Затем формируется массив объектов, который и будет отправлен на клиент:

*obj = {*

*token: item.token,*

*value: item.value,*

*age: item.age,*

*gender: item.gender,*

*date: item.date,*

*height: item.height,*

*weight: item.weight,*

*physicalActivity: item.physicalActivity,*

*alcohol: item.alcohol,*

*smoke: item.smoke,*

*normalValue: normalValue,*

*};*

*obj.rejectionReasons = apiController.analyzeRejectionReasons(obj);*

*array.push(obj);*

*});*

*res.send(array);*

*} else {next();};*

*});*

*});*

Ниже представлена функция, анализирующая записи истории пользователя и формирующая массив рекомендаций и диагнозов пользователя:

*let reasonsArray = [];*

*let systolicPressureValue = obj.value.split('/')[0];*

*let dyastolicPressureValue = obj.value.split('/')[1];*

*let systolicNormalPressureValue = obj.normalValue.split('/')[0];*

*let dyastolicNormalPressureValue = obj.normalValue.split('/')[1];*

*if (systolicPressureValue && dyastolicPressureValue) {*

Ниже представлена часть функции, анализирующая введенные пользователеи значения веса и роста:

*if (obj.height && obj.weight) {*

*let weight = obj.weight / (obj.height / 100 \* obj.height / 100);*

*if (weight < 18.5) {*

*reasonsArray.push('Recommend to gain weight');*

*} else if (weight > 25 && weight < 30) {*

*reasonsArray.push('Recommend to lose weight');*

*} else if (weight > 30) {*

*reasonsArray.push('Extremely recommend to lose weight');*

*}*

*}*

Следующий код отвечает за анализ введенного типа активности пользователя:

*if (obj.physicalActivity) {*

*if (obj.physicalActivity === 'notactive') {*

*reasonsArray.push('Try to increase your physical activity');*

*}}*

Код ниже отвечает за анализ введенного пользователем типа употребления алкоголя:

*if (obj.alcohol) {*

*if (obj.alcohol === 'normal') {*

*reasonsArray.push('Reduction in the amount of alcohol consumed can contribute to better health');*

*} else if (obj.alcohol === 'active') {*

*reasonsArray.push('You need to reduce the amount of alcohol consumed.');*

*}}*

Ниже происходит анализ введенного отношения пользователя к курению:

*if (obj.smoke) {*

*if (obj.smoke === true) {*

*reasonsArray.push('Smoking can cause pressure deterioration');*

*}}*

Ниже происходит анализ введенных пользователем значений систолического и дистолического давления:

*if (systolicNormalPressureValue && dyastolicNormalPressureValue) {*

*if (*

*(systolicNormalPressureValue - systolicPressureValue) > 20 &&*

*(dyastolicNormalPressureValue - dyastolicPressureValue) > 20*

*) {*

*reasonsArray.push('Try to avoid stress conditions');*

*reasonsArray.push('Try to avoid taking various stimulating substances');*

*}}*

*if (systolicPressureValue > 140) {*

*reasonsArray.push('There is a risk of cardiovascular disease.');*

*}*

Следующий код анализирует разницу между систолическим и дистолическим значениями давления пользователя:

*if ((systolicPressureValue - dyastolicPressureValue) > 55) {*

*reasonsArray.push('You have a significant difference between the upper and lower pressure, which may indicate a malfunction of the digestive system, lesions of the gallbladder, ducts, and tuberculosis.');*

*} else if ((systolicPressureValue - dyastolicPressureValue) < 30) {*

*reasonsArray.push('You have a lower difference in upper and lower pressure, which can lead to hypoxia, atrophic changes in the brain, impaired vision, respiratory paralysis, cardiac arrest. Seek medical attention immediately.');*

*}}*

Результатом работы модуля служит массив рекомендаций, сформированный в соответствии с входными значениями:

*return reasonsArray;*

После чего проанализированные данные отправляются на клиент, который в свою очередь приступает к их отображению. Для визуализации графика отклонений артериального давления от нормативных значений с течением времени использовалась библиотека *react-chartjs-2*, которая имеет широкий спектр геометрических возможностей.

## Модуль отображения выходных данных

Данный модуль не имеет какой-либо смысловой нагрузки с точки зрения логики, за исключением функции формирования графика:

*prepareDataForSchedule() {*

*let labelsArray = [];*

*let systolicPressureValues = [];*

*let dyastolicPressureValues = [];*

*let systolicNormalPressureValues = [];*

*let dyastolicNormalPressureValues = [];*

*this.state.history.forEach(function(item) {*

*labelsArray.push(item.date);*

*systolicPressureValues.push(item.value.split('/')[0]);*

*dyastolicPressureValues.push(item.value.split('/')[1]);*

*systolicNormalPressureValues.push(item.normalValue.split('/')[0]);*

*dyastolicNormalPressureValues.push(item.normalValue.split('/')[1]);*

*});*

*return {*

*labels: labelsArray,*

*datasets: [*

*{*

*label: 'systolic blood pressure',*

*data: systolicPressureValues*

*},{*

*label: 'diastolic blood pressure',*

*data: dyastolicPressureValues*

*},{*

*label: 'systolic normal blood pressure',*

*data: systolicNormalPressureValues*

*},{*

*label: 'diastolic normal blood pressure',*

*data: dyastolicNormalPressureValues*

*}*

*]};*

*}*

Он также разбирает пришедшие от сервера данные:

*if (this.state.history) {*

*reasons = this.state.history[this.state.history.length-1].rejectionReasons;*

*data = this.prepareDataForSchedule();*

*items = this.state.history.map(item => <li>{item.value} (Normal value:{item.normalValue})</li>);*

*rejectionReasons = <ul>{reasons.map(item => <li>{item}</li>)}</ul>*

*}*

Результатом работы модуля служит текстовое и визуальное отображение выходных данных.

Данный модуль изображен на рисунке 3.8.

## Модуль рассылки уведомлений

Данный модуль предназначен для рассылки уведомлений о изменениях атмосферного давления, которые могут повлиять на самочувствие пользователей. Вся логика данного модуля размещена на сервере, так как необходимо постоянное функционирование данного модуля и постоянная связь с базой данных. Для формирования и рассылки писем используется библиотка *Nodemailer.* Для начала происходит установка таймера, который будет вызывать функцию колбэк раз в сутки в определенное время:

*setInterval(function () {*

*var date = new Date();*

*if (date.getHours() === 7 && date.getMinutes === 0) {*

*apiController.sendNotifications()*

*};*

*}, 3600000)*

Функция колбэк содержит следующий код:

*sendNotifications() {*

*usersModel.find((error, users) => {*

*users.forEach(user => {*

*if(user.username.match( /@/ig ) && user.city) {*

*let testAccount = await nodemailer.createTestAccount();*

*let xhr = new XMLHttpRequest();*

*xhr.open('GET', 'http://api.apixu.com/v1/current.json?key=d9fe708ea76e4ad6977111820191705&q=' + user.city);*

*xhr.onreadystatechange = (resp) => {*

*let pressureValue = resp.condition.pressure\_mb / 133.466;*

*let text;*

*if (pressureValue > 770 || pressureValue < 740) {*

*text = 'Be careful, there is increased pressure, with high atmospheric pressure, weather-sensitive people should behave more passively, avoid physical exertion, measure blood pressure more often and consult a doctor if necessary. Additional medications should not be taken, however, with an increase in blood pressure, you can resort to short-acting drugs.';*

*} else if (pressureValue < 740) {*

*text = 'Be careful, there is a reduced atmospheric pressure, with low atmospheric pressure, weather-sensitive people should behave more passively, avoid physical exertion, measure blood pressure more often and, if necessary, consult a doctor. Additional medications should not be taken, however, with an increase in blood pressure, you can resort to short-acting drugs.';*

*}*

*if (text) {*

*let transporter = nodemailer.createTransport({*

*host: "smtp.ethereal.email",*

*port: 587,*

*secure: false,*

*auth: {*

*user: testAccount.user,*

*pass: testAccount.pass*

*}*

*});*

*await transporter.sendMail({*

*from: 'mimohojij@mail-finder.net',*

*to: user.username,*

*subject: "weather",*

*text: text,*

*});*

*}*

*};*

*xhr.send(JSON.stringify(data));*

*}*

*});*

*});*

*}*

## Руководство пользователя

Рассмотрим работу программного комплекса на рабочем примере. Рассмотрим пример работы для авторизованного пользователя. Первоначально пользователь попадает на главный экран неавторизованного пользователя, что отражено на рисунках 3.5 и 3.6.



Рисунок 3.5 – Форма определения нормативного давления



Рисунок 3.6 – Форма определения нормативного давления с выходными данными

Затем пользователю необходимо авторизоваться, либо зарегистрироваться если аккаунт отсутствует и затем авторизоваться, что изображено на рисунке 3.4. Данные шаги отражены на рисунках 3.2 и 3.3.



Рисунок 3.3 – Форма регистрации новых пользователей с сообщением о ошибке



Рисунок 3.2 – Форма регистрации новых пользователей



Рисунок 3.4 – Форма авторизации пользователей

Если авторизация прошла успешно, то пользователь попадает в личный кабинет, где имеется форма для введения данных. Данная форма представлена на рисунке 3.7.



Рисунок 3.7 – Модуль анализа значений артериального давления

После введения данных пользователь отправляет данные на сервер, затем полученный от сервера ответи визуализируется и пользователь может просмотреть полученные данные. Визуализация результирующих данных представлена на рисунке 3.8.



Рисунок 3.8 – Модуль отображения выходных данных

Затем чтобы выйти из системы пользователю необходимо вернуться на начальную страницу приложения и нажать кнопку *logout*.

# ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБСНОВАНИЕ РАЗРАБОТКИ И ИСПОЛЬЗОВАНИЯ АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ УПРАВЛЕНИЯ ПОЛОЧНЫМ ПРОСТРАНСТВОМ СУПЕРМАРКЕТА

## Характеристика автоматизированной системы управления полочным пространством супермаркета

Автоматизированная система управления полочным пространством супермаркета представляет собой систему, направленную на автоматизацию работы с товарами, находящимися на полках супермаркета. Система предоставляет возможность анализа данных о продуктах, находящихся на полках и поступающих на них, в результате чего выдаются конкретные параметры оптимального количества товаров, необходимых для наиболее прибыльной работы супермаркета, а также границы количества товаров, за пределами которых система оповещает о возможных потерях прибыли супермаркета в случае недостатка количества товаров на полках либо его избытке. Кроме того, система дает возможность оповещать об истекшем сроке годности. Данная система также позволяет хранить статистические данные о работе супермаркета в каждый конкретный период, на основе чего она может выдавать рекомендуемые варианты расположения и количества товаров в определенные дни.

Разработка и внедрение данной системы позволит:

* Исключить или уменьшить человеческий фактор при заполнении и последующем наблюдении за полками;
* Автоматизировать процесс расчета оптимального количества товара на полках, а также процесс расчета граничных значений количества товаров;
* Выполнять расчет необходимых значений количества товаров на определенный период времени;
* Автоматизировать процесс подсчета количества поступающих на полки и проданных товаров.

Целесобразность инвестиций в разработку, внедрение и использование автоматизированной системы заключается в сокращении расходов на анализ данных о товарах на полках супермаркета, а также в сокращении затрат на персонал в связи с автоматизацией процессов мерчендайзинга и наблюдения за товарами.

В результате разработки и внедрения данной автоматизированной системы появится возможность автоматизации процессов, связанных с работой с полочным пространством супермаркета, а также процессов сбора и анализа данных о поступивших и проданных товарах.

## 4.2 Расчет инвестиций в разработку автоматизированной системы

Расчет затрат на основную заработную плату команды разработчиков осуществляется исходя из состава и численности команды, размера месячной заработной платы каждого участника команды, а также трудоемкости работ, выполняемых при разработке программного средства отдельными исполнителями по формуле 4.1:

|  |  |
| --- | --- |
|  | (4.1) |

|  |  |  |  |
| --- | --- | --- | --- |
| где | Кпр | – | коэффициент премий (по данным предприятия или в диапазоне 1,5–2); |
|  | *n* | – | категории исполнителей, занятых разработкой программного средства; |
|  | Зч*i* | – | часовая заработная плата исполнителя *i*-й категории, р.; |
|  | *ti* | – | трудоёмкость работ, выполняемых исполнителем *i*-й категории, определяется исходя из сложности разработки программного обеспечения и объёма выполняемых им функций, ч. |

Коэффициент премий берется равным 1,5. Часовая заработная плата каждого исполнителя определяется путём деления его месячной заработной платы (оклад плюс надбавки) на количество рабочих часов в месяце, то есть на 168 часов.

При расчете заработной платы берется среднемесячная заработная плата в Республике Беларусь для сотрудников различных категорий ИТ-отрасли, поэтому премия не рассчитывается. Расчёт затрат на основную заработную плату приведен в таблице 4.1.

Дополнительная заработная плата разработчиков рассчитывается по формуле 4.2:

|  |  |
| --- | --- |
|  | (4.2) |

|  |  |  |  |
| --- | --- | --- | --- |
| где | Нд | – | норматив дополнительной заработной платы, 10 %. |

Таблица 4.1. – Расчет затрат на основную заработную плату команды разработчиков

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Категория исполнителя | Месячная заработная плата, р. | Часовая заработная плата, р. | Трудоемкость работ, ч | Итого, р. |
| 1 | 2 | 3 | 4 | 5 |
| Бизнес-аналитик | 3476,79 | 20,7 | 52,66 | 1090,06 |
| Системный архитектор | 7726,2 | 45,99 | 130,52 | 6002,62 |
| Программист | 4378,18 | 26,06 | 133,14 | 3469,63 |
| Тестировщик | 2575,4 | 15,33 | 97,59 | 1496,05 |
| Итого | 18156,57 | 108,08 | 413,88 | 12058,36 |
| Всего затраты на основную заработную плату разработчиков | | | | 18087,54 |

Отчисления на социальные нужды рассчитываются по формуле 4.3:

|  |  |
| --- | --- |
|  | (4.3) |

|  |  |  |  |
| --- | --- | --- | --- |
| где | Нсоц | – | ставка отчислений в ФСЗН и Белгосстрах, 34,6 %. |

Сумма прочих расходов рассчитывается по формуле 4.4:

|  |  |
| --- | --- |
|  | (4.4) |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| где | Нпр | – | норматив прочих расходов, 35 %. |

Общая сумма затрат на разработку рассчитывается по формуле 4.5:

|  |  |
| --- | --- |
|  | (4.5) |

Плановая прибыль, включаемая в цену программного средства рассчитывается по формуле 4.6:

|  |  |
| --- | --- |
|  | (4.6) |

|  |  |  |  |
| --- | --- | --- | --- |
| где | Рпс | – | рентабельность затрат на разработку программного средства, 30 %. |

Отпускная цена программного средства рассчитывается по формуле 4.7:

|  |  |
| --- | --- |
|  | (4.7) |

Формирование цены на основе затрат приведено в таблице 4.2.

Таблица 4.2. – Формирование цены программного средства на основе затрат

|  |  |  |
| --- | --- | --- |
| Наименование статьи затрат | Расчет по формуле (в таблице) | Значение, р. |
| 1 | 2 | 3 |
| 1 Основная заработная плата разработчиков | См. табл. 4.1 | 18087,54 |
| 2 Дополнительная заработная плата разработчиков |  | 1808,75 |
| 3 Отчисления на социальные нужды |  | 6884,12 |
| 4 Прочие расходы |  | 6330,64 |
| 5 Общая сумма затрат на разработку |  | 33111,05 |
| 6 Плановая прибыль, включаемая в цену программного средства |  | 9933,32 |
| 7 Отпускная цена программного средства |  | 43044,37 |

## 4.3 Расчет результата разработки и использования автоматизированной системы

Экономический эффект от разработки программного средства по индивидуальному заказу может быть рассчитан как для организации-разработчика, так и для организации-заказчика.

1. **Расчет экономического эффекта для организации-разработчика**

Для организации-разработчика экономическим эффектом является прирост чистой прибыли, полученной от разработки и реализации программного средства заказчику, который рассчитывается по формуле 4.8:

|  |  |
| --- | --- |
|  | (4.8) |

|  |  |  |  |
| --- | --- | --- | --- |
| где | Ппс | – | прибыль, включаемая в цену программного средства, р.; |
|  | Нп | – | ставка налога на прибыль, 18 %. |

Таким образом, экономический эффект для организации-разработчика составит:

1. **Расчет экономического эффекта для организации-заказчика**

Для организации-заказчика экономическим эффектом в результате использования разрабатываемой автоматизированной системы является прирост чистой прибыли, полученный за счет:

* экономии затрат на заработную плату с начислениями на заработную плату служащих в связи с сокращением их численности;
* экономии затрат на оплату труда и прочих затрат в результате снижения влияния человеческого фактора;
* снижения затрат на заработную плату с начислениями на заработную плату основных производственных рабочих.

Экономия на заработной плате и начислениях на заработную плату сотрудников за счет снижения трудоемкости работ рассчитывается по формуле 4.9:

|  |  |
| --- | --- |
|  | (4.9) |

|  |  |  |  |
| --- | --- | --- | --- |
| где | Кпр | – | коэффициент премий, 1,3; |
|  |  | – | трудоемкость выполнения работ сотрудниками до внедрения автоматизированной системы, ч; |
|  |  | – | трудоемкость выполнения работ сотрудниками после внедрения автоматизированной системы, ч; |
|  | Тч | – | часовая тарифная ставка сотрудника, использующего автоматизированную систему, р.; |
|  | *N*п | – | плановый объем работ, выполняемых сотрудником; |
|  | Нд | – | норматив дополнительной заработной платы, 15 %; |
|  | Нно | – | ставка отчислений от заработной платы, включаемых в себестоимость, 34,6 %. |

Автоматизированная система предполагает снижение трудозатрат для такой категории рабочего персонала, как фасовщики-комплектовщики. Поэтому, исходя из того, что в супермаркете работает три фасовщика-комплектовщика, = 6048 часов в год, = 4536 часов в год, Тч = 5 рублей, *N*п = 1. Соответственно экономия составит:

Экономия на заработной плате и начислениях на заработную плату в результате сокращения численности работников рассчитывается по формуле 4.10:

|  |  |
| --- | --- |
|  | (4.10) |

|  |  |  |  |
| --- | --- | --- | --- |
| где | Кпр | – | коэффициент премий, 1,3; |
|  | *n* | – | категории работников, высвобождаемых в результате внедрения программного средства; |
|  | *∆*Ч*i* | – | численность работников *i*-й категории, высвобожденных после внедрения программного средства, чел.; |
|  | З*i* | – | годовая заработная плата высвобожденных работников *i*-й категории после внедрения программного средства, р.; |
|  | Нд | – | норматив дополнительной заработной платы, 15 %; |
|  | Нсоц | – | норматив отчислений от заработной платы в соответствии с законодательством, 34,6 %. |

С учетом вводимых новшеств, автоматизированная система позволит сэкономить на заработной плате при увольнении мерчендайзера и контролера торгового зала. Таким образом, если после введения автоматизированной системы годовая заработная плата мерчендайзеров будет составлять 11400 рублей, а контролера торгового зала – 6600 рублей, экономия составит:

Соответственно экономия на текущих затратах составит:

Прирост чистой прибыли расчитывается по формуле 4.11:

|  |  |
| --- | --- |
|  | (4.11) |

|  |  |  |  |
| --- | --- | --- | --- |
| где |  | – | прирост текущих затрат, связанных с использованием автоматизированной системы, р. |

Для работы автоматизированной системы необходимо наличие интернета. Для этого производится подключение к интернет-провайдеру, услуги которого будут стоить 462 рубля в год. Кроме того, требуется своевременное обновление и решение проблем, возникающих в процессе работы системы. На поддержку и сопровождение будут выделяться средства в размере 40 % от итоговой стоимости системы, что составляет 17217,75 рубля. Затраты, связанные с использованием автоматизированной системы, составят:

Следовательно, прирост чистой прибыли составит:

## Расчет показателей экономической эффективности разработки и использования автоматизированной системы

Экономическая эффективность может быть рассчитана как для организации-разработчика, так и для организации-заказчика.

1. **Расчет показателей экономической эффективности разработки и использования автоматизированной системы для организации-разработчика**

Для организации-разработчика автоматизированной системы оценка экономической эффективности разработки осуществляется с помощью расчета простой нормы прибыли по формуле 4.12:

|  |  |
| --- | --- |
|  | (4.12) |

|  |  |  |  |
| --- | --- | --- | --- |
| где |  | – | прирост чистой прибыли, полученной от разработки автоматизированной системы организацией-разработчиком по индивидуальному заказу, р.; |
|  | Зр | – | затраты на разработку автоматизированной системы организацией-разработчиком, р. |

Таким образом, экономическая эффективность составит:

1. **Расчет показателей экономической эффективности разработки и использования автоматизированной системы для организации-заказчика**

Для организации-заказчика автоматизированной системы рассчитывается несколько показателей экономической эффективности, так как сумма инвестиций больше суммы годового экономического эффекта.

Прежде всего, для приведения доходов и затрат к настоящему моменту времени определяется коэффициент дисконтирования, который можно найти по формуле 4.13:

|  |  |
| --- | --- |
|  | (4.13) |

|  |  |  |  |
| --- | --- | --- | --- |
| где | Eн | – | требуемая норма дисконта, которая по своей природе соответствует норме прибыли, устанавливаемой инвестором в качестве критерия рентабельности инвестиций, доли единицы; |
|  | *t* | – | порядковый номер года, доходы и затраты которого приводятся к расчетному году; |
|  | *t*р | – | расчетный год, к которому приводятся доходы и инвестиционные затраты (*t*p = 1). |

Норма дисконта принимается равной 0,09. Расчетный период составит 4 года. Таким образом, коэффициенты дисконтирования за каждый год составят:

Также надо учесть, что в первый год экономический эффект будет значительно меньше планируемого, так как в течение этого года осуществляется разработка системы. Для того, чтобы учесть этот факт, необходимо выяснить, сколько времени будет затрачено на разработку системы. В данном случае необходимо отталкиваться от наибольшего количества часов, затрачиваемых на разработку. Проанализировав таблицу 4.1 можно увидеть, что больше всего трудозатраты у программиста: он затратит на разработку 133,14 часов. Так как при работе над системой программист отвечает не за все задачи, к данному времени необходимо добавить трудозатраты на такие этапы разработки, как планирование, анализ разрабатываемой системы, а также анализ и решение возникших при разработке проблем. Учитывая, что программист может работать над системой параллельно с определенными этапами, выявленные трудозатраты составят 35 часов. Поэтому общие трудозатраты на систему составят 168,14 часов, что равно одному рабочему году. Таким образом, выясняется, что в первый рассчитываемый год экономический эффект будет составлять 0 рублей.

Чистый дисконтированный доход – сумма дисконтированных значений потока платежей от проекта, приведенных к настоящему моменту времени. Он рассчитывается по формуле 4.14:

|  |  |
| --- | --- |
|  | (4.14) |

|  |  |  |  |
| --- | --- | --- | --- |
| где | Р*t* | – | экономический эффект в году *t*, р.; |
|  | З*t* | – | затраты (инвестиции) в году *t*, р.; |
|  | α*t* | – | коэффициент дисконтирования, рассчитанный для года *t*; |
|  | *n* | – | расчетный период, количество лет. |

Расчет чистого дисконтированного дохода за расчетный период приведен в таблице 4.3.

Таблица 4.3 – Расчёт эффективности инвестиций в разработку автоматизированной системы по индивидуальному заказу и его использование

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Показатель | Значение по годам расчетного периода | | | |
| 1 | 2 | 3 | 4 |
| **Результат** |  |  |  |  |
| 1. Прирост чистой прибыли, р. | 0 | 27766,44 | 27766,44 | 27766,44 |
| 2. Дисконтированный результат, р. | 0 | 25545,13 | 23323,81 | 21380,16 |
| **Затраты** |  |  |  |  |
| 3. Инвестиции в разработку автоматизированной системы, р. | 33111,05 | - | - | - |
| 4. Дисконтированные инвестиции, р. | 33111,05 | - | - | - |
| 5. Чистый дисконтированный доход по годам, р. | -33111,05 | 25545,13 | 23323,81 | 21380,16 |
| 6. Чистый дисконтированный доход нарастающим итогом, р. | -33111,05 | -7565,92 | 15757,89 | 37138,05 |

Продолжение таблицы 4.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Коэффициент дисконтирования, доли единицы | 1 | 0,92 | 0,84 | 0,77 |

Дисконтированный срок окупаемости – период возврата денежных средств с учетом временной стоимости денег. В отличие от срока окупаемости, данная величина приводит будущие денежные поступления к настоящему времени. Дисконтированный срок окупаемости будет равен моменту, когда суммарный дисконтированный результат превысит дисконтированную сумму инвестиций.

В данном случае дисконтированный эффект нарастающим итогом превысит дисконтированные инвестиции на третий год, поэтому дисконтированный срок окупаемости составит три года.

Рентабельность инвестиций без учета фактора времени рассчитывается по формуле 4.15:

|  |  |
| --- | --- |
|  | (4.15) |

|  |  |  |  |
| --- | --- | --- | --- |
| где | Пчср | – | среднегодовая чистая прибыль; |

Следовательно, рентабельность для организации-заказчика составит:

Срок окупаемости инвестиций определяется по формуле 4.16:

|  |  |
| --- | --- |
|  | (4.16) |

|  |  |  |  |
| --- | --- | --- | --- |
| где | Рср | – | среднегодовая сумма результата (экономического эффекта), р. |

Инвестиции вносятся в первый год разработки, то есть З1 = 33111,05 р., З2 = З3 = З4 = 0 р. Среднеговодая сумма результата составляет 20824,83 р.

Таким образом, срок окупаемости инвестиций составит:

То есть автоматизированная система окупится за 1 год и 7 месяцев.

## Выводы об экономической эффективности и целесообразности инвестиций

При разработке данной автоматизированной системы экономический эффект для организации-разработчика составит 8145,32 рубля. Кроме того, экономическая эффективность будет 24,6 %.

При внедрении автоматизированной системы для организации-заказчика расходы предприятия снизятся на 51541,26 рубля в год. Данная экономия связана с сокращением трудоемкости выполняемых работ персоналом и увольнением некоторых работников. Вложенные инвестиции окупятся чуть более чем за полтора года, а ежегодный прирост чистой прибыли составит 27766,44 рубля. Рентабельность инвестиций составит 62,89 %.

На основе данных показателей, можно сделать вывод, что внедрение автоматизированной системы полностью целесообразно.

**ЗАКЛЮЧЕНИЕ**

В процессе работы над дипломным проектом был разработан программный комплекс, позволяющий анализировать артериальное давление, предназначенный для помощи людям, имеющим проблемы с артериальным давлением, либо просто следящи за состоянием собственного здоровья.

В процессе работы над дипломным проектом были найдены аналоги в сфере анализа артериального давления и проведен их анализ. Были определены их сильные и слабые стороны.

В ходе выполнения дипломной работы были выполнены следующие задачи:

* + - разработать структуру программного комплекса анализа артериального давления;
    - спроектировать базу данных;
    - разработать алгоритмы функционирования программного комплекса;
    - разработать интерфейс в соответствии с требованиями эргономики;
    - реализовать программный продукт;
    - провести технико-экономическое обоснование разработки проекта.

Также были реализованы следующие функции автоматизированной системы:

* регистрация и авторизация пользователей;
* ввод данных;
* анализ введенных данных о артериальном давлении;
* вывод результатов анализа.

Программный комплекс представляет из себя клиент-серверное приложение, анализирующее входные данные артериального давления и сохранящее данные о пользователях и введенными ими данными в базе данных.

Было проведено технико-экономическое обоснование разработки программного комплекса, которое показало целесообразность его разработки с экономической и социальной точки зрения.

Цели, поставленные в дипломном проекте, были достигнуты в полном размере.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Виды и типы магазинов розничной торговли [Электронный ресурс]. – Режим доступа : https://litstile.ru/index.php?ht=1674.

[2] Супермаркет [Электронный ресурс]. – Режим доступа : https://znaytovar.ru/s/Supermarket.html.

[3] Что такое гипермаркет [Электронный ресурс]. – Режим доступа : https://znaytovar.ru/s/chto-takoe-gipermarket.html.

[4] Автоматизируем магазин продуктов [Электронный ресурс]. – Режим доступа : https://cloudshop.ru/content/avtomatiziruem-magazin-produktov.

[5] Терминал сбора данных [Электронный ресурс]. – Режим доступа : https://shtrih-m-spb.ru/catalog/tsd/.

[6] Классификация сканеров штрих-кода [Электронный ресурс]. – Режим доступа : https://posmagazin.ru/articles/klassifikatsiya-skanerov-shtrikh-koda/.

[7] Система управления полочным пространством магазина, автоматизация управления выкладкой [Электронный ресурс] – Режим доступа : https://abmcloud.com/abm-soft/shelf/.

[8] RS.ShelfSpace: система управления полочным пространством магазина, автоматизация мерчандайзинга, контроль выкладки товаров [Электронный ресурс] – Режим доступа : https://shelfspace.ru/.

[228] Зарплата в ИТ [Электронный ресурс] – Режим доступа : https://salaries.dev.by/.

[322] Работа в Минске, свежие вакансии [Электронный ресурс] – Режим доступа : https://jobs.tut.by/.

[

[8] Экономическое обоснование проекта по разработке программного обеспечения: методическое пособие. В. Г. Шоровой. Минск : БГУИР, 2018.

[9] Батин, Н. В. Дипломное проектирование. Методическое пособие / Н. В. Батин, А. А. Навроцкий – Минск : БГУИР, 2018. – 65 с.

[10] Документация Node.js [Электронный ресурс] – Электронные данные. – Режим доступа : https://nodejs.org/

[11] Документация Express.js [Электронный ресурс] – Электронные данные. – Режим доступа : https://expressjs.com/

[12] 25-я центральная районная поликилиника города Минска [Электронный ресурс] – Электронные данные. – Режим доступа : http://www.25crp.by

[13] Правила измерения артериального давления [Электронный ресурс] – Электронные данные. – Режим доступа : belmapo.by

[14] Ломако, А. В. Электронный учебно-методический комплекс по дисциплине «Проектирование автоматизированных систем» / А. В. Ломако – Минск : БГУИР, 2016.

[15] Когаловский М.Р. Энциклопедия технологий баз данных/ М.Р. Когаловский – Москва : Финансы и статистика, 2002. – 800 с.

Приложение А  
(обязательное)  
Листинг моделей соединения с базой данных

*let UserSchema = new mongoose.Schema({*

*email: {*

*type: String,*

*unique: true,*

*index: true,*

*required: true*

*},*

*password: {*

*type: String,*

*required: true*

*},*

*city: {*

*type: String,*

*}*

*});*

*let pressureSchema = new mongoose.Schema({*

*token: {*

*type: String*

*},*

*value: {*

*type: String*

*},*

*age: {*

*type: Number*

*},*

*gender: {*

*type: String*

*},*

*date: {*

*type: Number*

*},*

*height: {*

*type: Number*

*},*

*weight: {*

*type: Number*

*},*

*physicalActivity: {*

*type: String*

*},*

*alcohol: {*

*type: String*

*},*

*smoke: {*

*type: Boolean*

*}*

*});*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Обозначение* | | | | | *Наименование* | *Дополнительные сведения* | | | | |
|  | | | | | *Текстовые документы* |  | | | | |
|  | | | | |  |  | | | | |
| *БГУИР ДП 1-53 01 02 01 010 ПЗ* | | | | | *Пояснительная записка* | *65 с.* | | | | |
|  | | | | | *Отзыв руководителя* |  | | | | |
|  | | | | | *Рецензия* |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | | *Графические документы* |  | | | | |
|  | | | | |  |  | | | | |
| *ГУИР.000000.001 ПД* | | | | | *Организационная структура* |  | | | | |
|  | | | | | *программного комплекса* |  | | | | |
|  | | | | | *артериального давления* | *Формат А1* | | | | |
|  | | | | |  |  | | | | |
| *ГУИР.000000.002 ПЛ* | | | | | *Алгоритм неавторизованнго* |  | | | | |
|  | | | | | *пользователя* | *Формат А1* | | | | |
|  | | | | |  |  | | | | |
| *ГУИР.000000.003 ПД* | | | | | *Алгоритм авторизованного* |  | | | | |
|  | | | | | *пользователя* | *Формат А1* | | | | |
|  | | | | |  |  | | | | |
| *ГУИР.000000.004 ПЛ* | | | | | *Диаграмма декомпозиции* | *Формат А1* | | | | |
|  | | | | |  |  | | | | |
| *ГУИР.000000.005 ПЛ* | | | | | *Диаграмма вариантов* |  | | | | |
|  | | | | | *использования* | *Формат А1* | | | | |
|  | | | | |  |  | | | | |
| *ГУИР.000000.006 ПЛ* | | | | | *Скриншоты работы* |  | | | | |
|  | | | | | *программы* | *Формат А1* | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  |  |  |  |  | *БГУИР ДП 1-53 01 02 06 013 Д1* | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| *Изм.* | *Л.* | *№ докум.* | *Подп.* | *Дата* | *Автоматизированная система управления полочным пространством супермаркета*  *Ведомость дипломного  проекта* | *Лит* | | | *Лист* | *Листов* |
| *Разраб.* | | *Колесников В.Г* |  |  |  | *Т* |  | *65* | *65* |
| *Провер.* | | *Арановский В.Л.* |  |  | *Кафедра ИТАС*  *гр. 620603* | | | | |
| *Т.контр.* | | *Боброва Т.С.* |  |  |
| *Н.контр.* | | *Ярмолик В.И.* |  |  |
| *Утв.* | | *Навроцкий А.А.* |  |  |