МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Филиал федерального государственного бюджетного образовательного учреждения высшего образования

«Национальный исследовательский университет «МЭИ» в г. Смоленске

Кафедра вычислительной техники

Направление: 09.03.01 «Информатика и вычислительная техника»

Курсовая работа

по предмету: «Технология программирования»

Тема**:** «Автоматизированная информационная система продажи

авиабилетов**»**

Студент \_\_\_\_ИВТ1-19\_\_\_ \_\_\_\_\_\_\_\_\_\_\_ \_Милославский С. А.\_

группа подпись фамилия И.О.

Руководитель \_\_к. т. н.\_\_\_\_\_ \_\_доцент\_\_ \_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_Федулов Я. А.\_\_\_

учен. степень должность подпись фамилия И.О.

Смоленск – 2021

АННОТАЦИЯ

Пояснительная записка содержит 30 листов печатного текста, 17 рисунков, 1 таблицу, список литературы из 15 наименований и 4 формулы.

Данный курсовой проект состоит из введения, трёх разделов: анализ технического задания, проектирование разрабатываемой системы и реализация разрабатываемой системы; заключения, списка используемых источников и приложения.

В первом разделе анализируется техническое здание курсового проекта и объясняется выбор средств разработки проекта.

Во втором разделе разбирается предметная область курсового проекта и для наглядного представления бизнес логики разрабатываемой системы строятся следующие виды диаграмм:

* функциональная диаграмма IDEF0;
* диаграмма переходов состояний STD;
* диаграмма вариантов использования;
* диаграмма деятельности;
* диаграмма последовательности действий;
* диаграмма Джексона.

Кроме этого во втором разделе приводятся схемы алгоритмов, отражающие выполнение и работу проекта.

В третьем разделе осуществляется сама реализация разрабатываемой системы. Строятся две диаграммы: классов и компонентов, кроме этого в этом разделе приводятся основные интерфейсные формы проекта, строится граф диалога и выполняется тестирование проекта двумя способами: структурное и функциональное тестирование.

Структурное тестирование включает в себя тестирование:

* базового пути;
* условий;
* циклов.

Функционально тестирование включает в себя:

* разбиение на классы эквивалентности;
* анализ граничных значений;
* анализ причинно-следственных связей.

Объект исследования: программные средства для продаж авиабилетов.

Цель работы: разработать автоматизированную информационную систему продажи авиабилетов.

Технология разработки – Visual Studio 2019, PostgreSQL server 13.0, C#

Результаты работы: спроектирована и разработана автоматизированная информационная система продажи авиабилетов.

Содержание

[ВВЕДЕНИЕ 6](#_Toc73628462)

[1 АНАЛИЗ ТЕХНИЧЕСКОГО ЗАДАНИЯ 8](#_Toc73628463)

[1.1 Выбор средств разработки 9](#_Toc73628464)

[1.2 Выводы по разделу 11](#_Toc73628465)

[2 ПРОЕКТИРОВАНИЕ РАЗРАБАТЫВАЕМОЙ СИСТЕМЫ 12](#_Toc73628466)

[2.1 Анализ предметной области 12](#_Toc73628467)

[2.2 Функциональная диаграмма IDEF0 13](#_Toc73628468)

[2.3 Диаграмма переходов состояний 20](#_Toc73628469)

[2.4 Функциональные возможности разрабатываемой системы 21](#_Toc73628470)

[2.5 Динамические аспекты поведения разрабатываемой системы. 24](#_Toc73628471)

[2.6 Интерактивное поведение разрабатываемой системы 27](#_Toc73628472)

[2.7 Диаграмма Джексона 29](#_Toc73628473)

[2.8 Схемы алгоритмов 30](#_Toc73628474)

[2.9 Выводы по разделу 30](#_Toc73628475)

[3 РЕАЛАЛИЗАЦИЯ РАЗРАБАТЫВАЕМОЙ СИСТЕМЫ 31](#_Toc73628476)

[3.1 Структурные или функциональные схемы 31](#_Toc73628477)

[3.1.1 Диаграмма компонентов 31](#_Toc73628478)

[3.1.2 Диаграмма классов 31](#_Toc73628479)

[3.2 Реализация пользовательского интерфейса 32](#_Toc73628480)

[3.3 Тестирование 32](#_Toc73628481)

[3.3.1 Структурное тестирование 32](#_Toc73628482)

[3.3.1.1 Тестирование базового пути 32](#_Toc73628483)

[3.3.1.2 Тестирование условий 33](#_Toc73628484)

[3.3.1.3 Тестирование циклов 33](#_Toc73628485)

[3.3.2 Функциональное тестирование 33](#_Toc73628486)

[3.3.2.1 Разбиение на классы эквивалентности 33](#_Toc73628487)

[3.3.2.2 Анализ граничных значений 33](#_Toc73628488)

[3.3.2.3 Анализ причинно-следственных связей 33](#_Toc73628489)

[3.4 Оценка качества разрабатываемой системы 34](#_Toc73628490)

[3.5 Выводы по разделу 34](#_Toc73628491)

[ЗАКЛЮЧЕНИЕ 35](#_Toc73628492)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 36](#_Toc73628493)

[ПРИЛОЖЕНИЕ А – КОД ПРОГРАММЫ 37](#_Toc73628494)

[ПРИЛОЖЕНИЕ Б – ДИСК С разработанной системой 38](#_Toc73628495)

ВВЕДЕНИЕ

В настоящее время информационные системы занимают значимое место в жизни людей. Самые первые из них были созданы еще в 50-х годах прошлого столетия и осуществляли в основном арифметические расчеты, незначительно сокращая издержки производства и затраты времени. Развитие информационных систем не стояло на месте, продвигаясь в ногу со временем и деловыми потребностями человека. К банальным возможностям расчета зарплат добавились возможности анализировать информацию, упрощая процесс принятия решений для управленческого персонала. Также, с каждым годом степень автоматизации систем увеличивалась, позволяя все сильнее наращивать производственные показатели предприятий их использующих.

В современных условиях, человек вынужден работать с гигантскими объемами информации. В связи с этим разработка программных продуктов, служащих для автоматизированного учета, весьма актуальна. Системы обязаны представлять собой мощные средства, способные обрабатывать гигантские потоки данных высокой структурной сложности за минимум затраченного времени, обеспечивая дружественный диалог с пользователем.

На данный момент существует огромное количество информационных систем, занимающихся продажами авиабилетов. Они могут представлять собой как самостоятельные приложения, так и онлайн сервисы, предоставляя пользователю доступ к веб-службам поставщиков.

Целью данной курсовой работы является создание автоматизированной информационной системы, осуществляющей продажи авиабилетов.

Разработка подобной системы весьма актуальна на данный момент. В современном мире самолеты являются не только самым быстрым видом транспорта, но и самым безопасным, в связи с этим авиаперелеты пользуются весьма высокой популярностью. Вследствие этого продаваемые на рейсы билеты востребованы и с высокой вероятностью найдут своего покупателя, при условии, что авиакомпания обеспечила клиенту полноценный доступ к нужной ему информации. Это и есть задача, решаемая современными автоматизированными информационными системами. Существует множество подобных разработок, позволяющих авиакомпаниям реализовывать авиабилеты, а пользователям приобретать их. Однако, зачастую, функциональность таких систем либо весьма ограничена, либо предоставляет достаточное количество информации, жертвуя дружественностью к пользователю.

Для реализации поставленной задачи были выбраны следующие средства разработки: MS Visual Studio 2019, PostgreSQL server 13.0 и язык программирования C#.

1 АНАЛИЗ ТЕХНИЧЕСКОГО ЗАДАНИЯ

Разрабатываемая система должна содержать в себе следующие подсистемы:

* подсистема администрирования, позволяющая осуществлять настройку системы и ее поддержку;
* клиентская подсистема, позволяющая просматривать справочную информацию и отправлять запросы на бронирование или возврат авиабилетов.

Для доступа к любой из данных подсистем пользователь должен пройти предварительную регистрацию или, если пользователь уже зарегистрирован, авторизацию. Разным группам пользователей доступны разные функциональные возможности и уровень доступа к информации.

В ходе данной курсовой работы должна быть создана автоматизированная информационная система продажи авиабилетов, решающая следующие задачи:

* продажа авиабилетов на запланированные рейсы;
* поиск авиабилетов по запросу пользователя;
* поиск подходящего рейса при поступлении заявки;
* администрирование информационной системы;
* создание приложения, предоставляющего пользователям графический интерфейс для доступа к системе.

Архитектурой информационной системы продажи авиабилетов называется концепция, определяющая структуру и взаимосвязь компонентов в данной системе. Компоненты информационной системы по выполняемым функциям можно разделить на три слоя: слой представления, слой бизнес-логики и слой доступа к данным. Разрабатываемой информационной системе продажи авиабилетов соответствует клиент-серверная архитектура (рисунок 1).



Рисунок 1 – Архитектура клиент-сервер

Такой подход обеспечивает многопользовательский режим доступа к информации, а также гарантирует целостность данных. Пользователь использует спроектированный графический интерфейс клиентской части системы, которая в свою очередь отправляет запросы серверному ПО. Также особенностью такой архитектуры является разделение функциональных возможностей приложения между клиентом и сервером.

1.1 Выбор средств разработки

Согласно техническому заданию необходимо разработать автоматизированная информационная система продажи авиабилетов. Для реализации такой системы были выбраны следующие средства для разработки: MS Visual Studio 2019, PostgreSQL server 13.0 и язык программирования C#.

Microsoft Visual Studio – это интегрированная среда разработки (IDE) от Microsoft. Он используется для разработки компьютерных программ, а также веб-сайтов, веб-приложений, веб-сервисов и мобильных приложений. Visual Studio включает редактор кода, поддерживающий IntelliSense (компонент завершения кода), а также рефакторинг кода. Интегрированный отладчик работает как отладчик на уровне исходного кода, так и как отладчик на уровне машины. Другие встроенные инструменты включают профилировщик кода, конструктор для создания приложений с графическим интерфейсом, веб-дизайнер, конструктор классов и конструктор схемы базы данных. Данная среда разработки используется из-за ее возможностей в области работы с пользовательским интерфейсом, рефакторинга кода и удобной отладки.

Microsoft Visio – это приложение для создания диаграмм и векторной графики, входящее в семейство Microsoft Office. Для разработки схем процессов не требуется специальное обучение. Построение диаграмм и схем процессов осуществляется с помощью простого и понятного интерфейса. Одним из основных преимуществ Microsoft Visio является большое количество различных шаблонов диаграмм, что упрощает и ускоряет процесс создания схем бизнес процессов (например, eEPC, IDEF0, IDEF3, UML). Данное приложение использовалось для построения диаграмм и схем в данном проекте.

PostgreSQL (Postgres) – система управления реляционными базами данных (RDBMS) с открытым исходным кодом, в которой особое внимание уделяется расширяемости и совместимости с SQL. Эта система предназначена для обработки и хранения данных. В данном проекте служит хранилищем данных.

pgAdmin – инструмент для администрирования данных на сервере PostgreSQL. Он представляет собой графический клиент для работы с сервером, через который можно с легкостью создавать, удалять, изменять базы данных и управлять данными. В проекте этот инструмент использовался для проверки и манипуляции данными.

Redis – хранилище структуры данных, используемое в качестве распределенной базы данных ключ-значение, кеша и брокера сообщений с дополнительной надежностью. В проекте используется для аутенфикации.

Windows Subsystem for Linux (WSL) – это уровень совместимости для запуска Linux приложений в ОС Windows 10. В проекте используется для удобной работы с Redis на Windows.

GraphQL – это язык запросов и обработки данных с открытым исходным кодом для API, а также среда выполнения для выполнения запросов с существующими данными.

1.2 Выводы по разделу

В этом разделе курсового проекта анализировалось техническое задание. Из приведенного анализа был спроектирован необходимый функционал для автоматизированной информационной системы продажи авиабилетов.

Кроме этого в этом разделе описан выбор инструментальных средств и разобраны основные средства компоненты, которые участвуют в реализации проекта.

2 ПРОЕКТИРОВАНИЕ РАЗРАБАТЫВАЕМОЙ СИСТЕМЫ

2.1 Анализ предметной области

Целью данной курсовой работы является создание программного средства, осуществляющего поиск и продажу авиабилетов.

Авиакомпания занимается авиаперевозками пассажиров. Также она устанавливает маршруты полетов. Рейсы осуществляются по установленным маршрутам согласно расписанию. На каждый рейс существует определенное количество билетов. Продажа билета пользователю осуществляется при отправке запроса на бронирование, при условии, что данный билет до сих пор есть в наличии. Приобретя билет, пользователь предоставляет информацию о себе и становится пассажиром. Совершеннолетние пассажиры обязаны иметь занесенные в базу данных (БД) паспортные данные. Несовершеннолетние обязаны иметь занесенные в БД данные из свидетельства о рождении. Администраторы системы могут ограничивать или расширять доступ пользователей и сотрудников к предоставляемой информации.

Система создаётся для обслуживания следующих групп пользователей:

* пользователи, приобретающие билеты на рейсы и осуществляющие их поиск;
* администраторы, осуществляющие контроль за пользователями и функциональностью системы.

Абсолютно каждая авиакомпания использует определенную систему дистрибуции. Наиболее развитые используют GDS (глобальные дистрибьюторские системы, которые формируются из основных международных компьютерных систем резервирования). В итоге сервисы продаж авиабилетов при поиске информации пользуются ресурсами глобальных дистрибьюторских систем. Однако доступ к GDS является не бесплатным, поэтому в роли дистрибутивной системы для разрабатываемой системы будет выступать БД, созданная в PostgreSQL server 13.0.

В БД должна храниться информация:

* о маршрутах;
* о рейсах;
* о пользователях;
* о билетах;
* об авиакомпаниях;
* о сотрудниках;
* о пассажирах.

Разрабатываемая информационная система предназначена для продажи авиабилетов и упрощения доступа к нужной информации. Наличие данной разработки улучшает организационную работу авиаперевозчика за счёт отсутствия бумажной документации, поиск и систематизация которой занимали бы очень большое количество времени

2.2 Функциональная диаграмма IDEF0

IDEF0 – нотация графического моделирования, используемая для создания функциональной модели, отображающей структуру и функции системы, а также потоки информации и материальных объектов, связывающих эти функции. Тем не менее методология IDEF0 является одним из популярных подходов для описания бизнес-процессов. К ее особенностям можно отнести:

* использование контекстной диаграммы;
* поддержка декомпозиции;
* доминирование;
* выделение 4 типов стрелок.

Контекстная диаграмма – самая верхняя диаграмма, на которой объект моделирования представлен единственным блоком с граничными стрелками (в каждой модели IDEF0 может быть только одна контекстная диаграмма). Эта диаграмма называется A-0 (А минус нуль). Стрелки на этой диаграмме отображают связи объекта моделирования с окружающей средой. Диаграмма A-0 устанавливает область моделирования и ее границу.

Нотация IDEF0 поддерживает последовательную декомпозицию процесса до требуемого уровня детализации. Дочерняя диаграмма, создаваемая при декомпозиции, охватывает ту же область, что и родительский процесс, но описывает ее более подробно. Согласно методологии, IDEF0 при декомпозиции стрелки родительского процесса переносятся на дочернюю диаграмму в виде граничных стрелок.

В данном пункте пояснительной записки приводится диаграмма IDEF0, которая разделена на три уровня иерархии диаграммы.

На рисунке 2.2.1 представлена контекстная диаграмма «Автоматизированная информационная система продажи авиабилетов», отражающая нулевой уровень иерархии. Из этой диаграммы видно, как разрабатываемая система взаимодействует с данными, и кто может к ней иметь доступ.

На рисунке 2.2.2 представлена диаграмма декомпозиции «Автоматизированная информационная система продажи авиабилетов», которая отражает второй уровень иерархии. Эта диаграмма представляет собой подробное описание контекстной диаграммы (рисунок 2.2.1). Из этой диаграммы видно, как разрабатываемая система взаимодействует с пользователями и какой доступ к данным они имеют.

Аналогичным образом строились диаграммы декомпозиции, представленные на рисунках 2.2.3–2.2.5, отражающие третий уровень иерархии. Из этих диаграмм видно, как данные взаимодействуют с БД и какие права имеют пользователи этой системы.



Рисунок 2.2.1 – Контекстная диаграмма «Автоматизированная информационная система продажи

авиабилетов»



Рисунок 2.2.2 – Диаграмма декомпозиция модуля «Автоматизированная информационная система продажи

авиабилетов»



Рисунок 2.2.3 – Диаграмма декомпозиция модуля «Аутенфикация в системе»



Рисунок 2.2.4 – Диаграмма декомпозиция модуля «Администрирование системы»



Рисунок 2.2.5 – Диаграмма декомпозиция модуля «Покупка авиабилета»

2.3 Диаграмма переходов состояний

Диаграммы переходов состояний (STD) предназначены для моделирования и документирования аспектов систем, зависящих от времени или реакции на событие. Они позволяют осуществлять декомпозицию управляющих процессов и описывают отношения между входными и выходными управляющими потоками для управляющего процесса-предка.

С помощью STD-диаграмм можно моделировать последующее функционирование системы на основе ее предыдущего и текущего функционирования. Моделируемая система в любой заданный момент времени находится точно в одном из конечного множества состояний. С течением времени она может изменить свое состояние, при этом переходы между состояниями должны быть точно определены. На рисунке 2.3.1 представлена диаграмма состояний для окна «Покупка авиабилета», отражающая состояние перехода процессов для интерфейсной формы «Покупка авиабилета».



Рисунок 2.3.1 – Диаграмма состояний для окна «Покупка авиабилета»

2.4 Функциональные возможности разрабатываемой системы

Функциональные возможности системы представлены в диаграмме вариантов использования (рисунок 2.4.1).

Диаграмма вариантов использования показывает отношения между актерами и прецедентами, описывая систему на концептуальном уровне. В данном случае существует девять актеров:

* пользователь;
* зарегистрированный пользователь;
* незарегистрированный пользователь;
* незарегистрированный сотрудник;
* клиент;
* кассир;
* диспетчер;
* работник отдела кадров;
* системный администратор.

Причем для предотвращения дублирования информации было использовано отношение обобщения.

Представленные на диаграмме варианты использования показывают с какими данными могут взаимодействовать актеры и какие действия они могут выполнять над этими данными. Кроме того, что варианты использования показывают действия актеров над данными, так они отражают своего рода права доступа актеров к предоставленным им данным в разрабатываемой системе.

Для каждого варианта использования должна строиться таблица, отражающая в себе следующую информацию:

* название;
* действующие лица
* цель;
* сценарий;
* результат.



Рисунок 2.4.1 – Диаграмма вариантов использования

В данном пункте пояснительной записки приведена часть таких таблиц.

В таблицах 2.4.1 и 2.4.2 описаны варианты использования «Регистрация» и «Войти в систему», представленные на рисунке 2.4.1.

Таблица 2.4.1 – Краткое описание варианта использования «Регистрация»

|  |  |
| --- | --- |
| Название | Регистрация |
| Действующие лица | Пользователь, «АТО» |
| Цель | Создать нового пользователя «АТО» |
| Предусловие | Нет |
| Успешный сценарий | 1. Пользователь запускает программу и видит окно входа в аккаунт. 2. Пользователь нажимает на кнопку «Создать аккаунт». 3. Программа открывает новое окно регистрации. Окно входа в систему скрывается. 4. Пользователь заполняет все необходимые поля: Логин, Пароль, Имя, Фамилия, Отчество. 5. Пользователь нажимает на кнопку «Регистрация». 6. Программа проверяет правильность ввода, и в случае успешной проверки данные пользователя добавляются в соответствующие таблицы БД.   Окно регистрации закрывается, и вновь отрывается окно входа в программу. |
| Результат | Пользователь добавлен в систему. |

Таблица 2.4.2 – Краткое описание варианта использования «Войти в систему»

|  |  |
| --- | --- |
| Название | Войти в систему |
| Действующие лица | Зарегистрированный пользователь, «АТО» |
| Цель | Войти в систему «АТО» |
| Предусловие | Пользователь ранее был зарегистрирован в системе «АТО» |
| Успешный сценарий | 1. Пользователь запускает программу и видит окно входа в аккаунт. 2. Пользователь вводит свой логин и пароль. 3. Пользователь нажимает на кнопку «Войти». 4. Программа проверяет наличие пользователя с таким логином и паролем, и в случае успешной проверки пользователь входит в свой аккаунт. 5. Окно входа в аккаунт срывается. 6. Отрывается окно с дополнительными возможностями (это окно зависит от типа пользователя). |
| Результат | Пользователь вошел в свой аккаунт «АТО» и получил дополнительные возможности в зависимости от прав доступа.. |

Построенные таблицы вариантов использования отражают подробное описание определенного варианта использования, изображенного на рисунке 2.4.1. Аналогичным образом строятся таблицы для остальных вариантов использования.

2.5 Динамические аспекты поведения разрабатываемой системы.

Динамические аспекты поведения системы представлены в диаграмме активности.

Диаграмма активности позволяет построить графическое представление потока от одного действия к другому. Действие можно описать как работу системы. Поток управления перенаправляется от одной операции к другой. Этот поток может быть:

* последовательным;
* разветвленным;
* параллельным.

Диаграммы активности в основном используются в качестве блок-схемы, которая состоит из действий, выполняемых системой. Она фиксирует динамическое поведение системы, иными словами используется для отображения потока сообщений от одного действия к другому.

Активности на диаграмме «разбросаны» по беговым дорожкам, каждая из которых соответствует поведению одного из объектов. Благодаря этому легко определить, каким из объектов выполняется каждая из активностей.

Дорожка – часть области диаграммы деятельности, на которой отображаются только те активности, за которые отвечает конкретный объект. Предназначены дорожки для разбиения диаграммы в соответствии с распределением ответственности за действия, выполняемых разными участниками в одном потоке. Имя дорожки может означать роль или объект, которому она соответствует.

На рисунках 2.5.1 и 2.5.2 представлены диаграммы активности «Войти в систему» и «Заказ авиабилета». Эти диаграммы отражают пошаговое выполнение действий вариантов использования представленных на рисунке 2.4.1. Кроме пошагово выполнения действий на этих диаграммах присутствуют дорожки «Пользователь» и «АТО», которые разбивают диаграмму на группировки действий. Представленное на диаграммах разбиение действий на группировки показывает какой объект отвечает за какие процессы выполнения действий.



Рисунок 2.5.1 – Диаграмма активности «Войти в систему»



Рисунок 2.5.2 – Диаграмма активности «Заказ авиабилета»

2.6 Интерактивное поведение разрабатываемой системы

Интерактивное поведение системы представлены в диаграмме последовательности.

Диаграмма последовательности показывает взаимодействие между объектами в последовательном порядке, то есть в порядке, в котором эти взаимодействия происходят. На ней изображаются объекты, задействованные в сценарии, и последовательность сообщений, которыми обмениваются объекты, необходимые для выполнения функций сценария. Диаграммы последовательностей обычно связаны с реализациями вариантов использования в логическом представлении разрабатываемой системы.

Диаграмма последовательности показывает в виде параллельных вертикальных линий (линий жизни) разные процессы или объекты, которые существуют одновременно, а в виде горизонтальных стрелок – сообщения, которыми они обмениваются, в том порядке, в котором они происходят. Это позволяет описывать простые сценарии выполнения в графическом виде.

На рисунках 2.6.1 и 2.6.2 представлены диаграммы последовательности «Авторизация в системе» и «Бронирование авиабилета». Эти диаграммы отражают взаимодействие пользователя с разрабатываемой системой. Из-за выбранной архитектуры клиент-сервер принцип работы на клиенте одинаков, в связи с этим диаграммы последовательности похожи друг на друга, кроме формы взаимодействия с пользователем.

По представленным диаграммам последовательности видно, как осуществляется выполнение запросов на клиенте, а именно как данные взаимодействует через изображенные объекты. Выполнение запросов происходит следующем образом.

Пользователь взаимодействует с формой, то есть заполняет форму необходимыми данными. Далее выполняется соответствующий запрос, который передает введенные пользователем данные на форме серверу. Сервер обрабатывает полученные данные и тоже вызывает соответствующий запрос, который обращается уже к БД. После обработки запроса в БД пользователь получает ответ на свои действия.



Рисунок 2.6.1 – Диаграмма последовательности «Авторизация в системе»



Рисунок 2.6.2 – Диаграмма последовательности «Бронирование авиабилета»

2.7 Диаграмма Джексона

Структурированное программирование Джексона (JSP) – это метод структурного программирования, разработанный британским консультантом по программному обеспечению Майклом А. Джексоном и описанный в его книге. Техника JSP заключается в анализе структур данных файлов, которые программа должна читать как ввод и создавать как вывод, а затем создавать проект программы на основе этих структур данных, чтобы структура управления программой обрабатывала эти структуры данных. естественным и интуитивно понятным способом. JSP описывает структуры (как данных, так и программ) с использованием трех основных структур:

* последовательность;
* итерация;
* выбор (или альтернативы).

Эти структуры представлены в виде (по сути) визуального представления регулярного выражения.

На рисунке 2.7.1 представлена диаграмма Джексона «Покупка авиабилета».



Рисунок 2.7.1 – Диаграмма Джексона

2.8 Схемы алгоритмов

2.9 Выводы по разделу

3 РЕАЛАЛИЗАЦИЯ РАЗРАБАТЫВАЕМОЙ СИСТЕМЫ

3.1 Структурные или функциональные схемы

3.1.1 Диаграмма компонентов

Диаграмма компонентов описывает организацию и подключение физических компонентов в системе, то есть показывает структурные отношения между компонентами системы. Диаграммы компонентов используется, чтобы помочь в деталях реализации модели и перепроверить, что каждый аспект требуемых функций системы охвачен запланированной разработкой.

В этой диаграмме компоненты являются менее физическими и более концептуальными автономными элементами дизайна, такими как бизнес-процесс, который предоставляет или требует интерфейсов для взаимодействия с другими конструкциями в системе. Физические элементы, такие как файлы и документы, теперь называются артефактами

3.1.2 Диаграмма классов

https://www.tutorialspoint.com/uml/uml\_class\_diagram.htm

https://developer.ibm.com/technologies/web-development/articles/the-class-diagram/

https://creately.com/blog/diagrams/class-diagram-relationships/

https://www.guru99.com/uml-class-diagram.html

https://en.wikipedia.org/wiki/Class\_diagram

Диаграмма классов – статическая структура, которая дает обзор программного обеспечения системы, отображая классы, атрибуты, операции и их отношения между собой. Эта диаграмма включает имя класса, атрибуты и операцию в отдельных обозначенных отсеках. Диаграмма классов помогает построить код для разработки программного обеспечения.

Диаграмма классов определяет типы объектов в системе и различные типы отношений, которые существуют между ними. Это дает общее представление о приложении. Этот метод моделирования может работать практически со всеми объектно-ориентированными методами. Класс может относиться к другому классу. Класс может иметь свои объекты или может наследовать от других классов.

3.2 Реализация пользовательского интерфейса

построение графа диалога)

https://studbooks.net/2016210/informatika/proektirovanie\_polzovatelskogo\_interfeysa

https://studbooks.net/2258632/informatika/postroenie\_grafa\_dialoga

https://studwood.ru/1884866/informatika/postroenie\_grafa\_dialoga

Граф диалога – ориентированный взвешенный граф, каждой вершине которого соответствует определенное состояние диалога, характеризующееся набором доступных пользователю действий. Дуги, исходящие из вершин, показывают возможные изменения состояний при выполнении пользователем указанных действий. Таким образом, граф представляет собой набор состояний системы, между которыми в ходе диалога при определенных условиях осуществляются переходы.

Разработка графа диалога позволяет выявить и устранить возможные тупиковые ситуации, выбрать рациональный путь перехода из текущего состояния системы в требуемое, выявить неоднозначные ситуации, когда для пользователя требуется дополнительная помощь.

Интерфейс пользователя можно упростить, снизив степень неопределенности действий пользователя. Для этого можно применить смешанную структуру диалога, ограничив при необходимости свободу выбора пользователя, используя меню или другие элементы и контролировать вводимую пользователем информацию, принимать только допустимые данные.

3.3 Тестирование

3.3.1 Структурное тестирование

https://www.softwaretestingclass.com/what-is-structural-testing/

Структурное тестирование – это тип тестирования, проводимого для проверки структуры кода. Это также известно, как тестирование «белого ящика» или «тестирование стеклянного ящика». Этот тип тестирования требует знания кода, поэтому в основном его проводят разработчики. Его больше заботит то, как это работает система, а не ее функциональные возможности. Это обеспечивает больший охват тестирования.

Структурные методы тестирования:

* тестирование базового пути;
* тестирование условий;
* тестирование циклов.

3.3.1.1 Тестирование базового пути

https://www.guru99.com/basis-path-testing.html

3.3.1.2 Тестирование условий

https://www.guru99.com/test-scenario-vs-test-condition.html

3.3.1.3 Тестирование циклов

https://www.guru99.com/loop-testing.html

3.3.2 Функциональное тестирование

https://www.guru99.com/functional-testing.html

Функциональное тестирование – это тип тестирования программного обеспечения, при котором проверяется соответствие системы программного обеспечения функциональным требованиям или спецификациям. Целью функциональных тестов является тестирование каждой функции программного приложения путем предоставления соответствующих входных данных и проверки выходных данных на соответствие функциональным требованиям.

Функциональное тестирование в основном включает в себя тестирование черного ящика и не касается исходного кода приложения. Это тестирование проверяет пользовательский интерфейс, API, базу данных, безопасность, связь клиент или сервер и другие функции тестируемого приложения. Тестирование можно проводить вручную или с помощью автоматизации.

3.3.2.1 Разбиение на классы эквивалентности

https://www.guru99.com/equivalence-partitioning-boundary-value-analysis.html

3.3.2.2 Анализ граничных значений

3.3.2.3 Анализ причинно-следственных связей

3.4 Оценка качества разрабатываемой системы

Оценка качества разрабатываемой системы выполняется согласно ГОСТ 28195-89. Этот стандарт, устанавливает общие положения по оценке качества программных средств вычислительной техники (ПС), поставляемых через фонды алгоритмов и программ (ФАП), номенклатуру и применяемость показателей качества ПС.

Согласно госту 28195-89 был выбран 5012 (программные средства управления базами данных) подкласс показателей качества ПС. Показатели качества этого подкласса приведены в таблице 3.4.1.

|  |  |  |
| --- | --- | --- |
| Наименование групп и комплексных показателей качества | Обозначение показателя | Применяемость показателя по подклассам (группам) ПС |
| 1. Показатели надежности ПС |  |  |
| 1.1. Устойчивость функционирования | H1 | + |
| 1.2. Работоспособность | Н2 | + |
| 2. Показатели сопровождения |  | + |
| 2.1. Структурность | С1 | + - |
| 2.2. Простота конструкции | С2 | + - |
| 2.3. Наглядность | С3 | + - |
| 2.4. Повторяемость | С4 | + - |
| 3. Показатели удобства применения |  |  |
| 3.1. Легкость освоения | У1 | + - |
| 3.2. Доступность эксплуатационных программных документов | У2 | + |
| 3.3. Удобство эксплуатации и обслуживания | УЗ | + |
| 4. Показатели эффективности |  | + |
| 4.1. Уровень автоматизации | Э1 | + - |
| 4.2. Временная эффективность | Э2 | + - |
| 4.3. Ресурсоемкость | Э3 | + |
| 5. Показатели универсальности |  | + |
| 5.1. Гибкость | Г1 | + - |
| 5.2. Мобильность | Г2 | + - |
| 5.3. Модифицируемость | Г3 | + |
| 6. Показатели корректности |  | + |
| 6.1. Полнота реализации | К1 | + |
| 6.2. Согласованность | К2 | + |
| 6.3. Логическая корректность | К3 | + |
| 6.4. Проверенность | К4 | + |

3.5 Выводы по разделу

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

ПРИЛОЖЕНИЕ А – КОД ПРОГРАММЫ

ПРИЛОЖЕНИЕ Б – ДИСК С разработанной системой