|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **СОДЕРЖАНИЕ**  ВВЕДЕНИЕ. 4  1. ПОСТАНОВКА ЗАДАЧИ. 6  1.1 Средства решения поставленной задачи. 8  1.2 Реализация базы данных 10  1.3 Запуск серверной части 14   1. РАЗРАБОТКА АЛГОРИТМОВ. 16    1. Алгоритмы общего задания. 16 2. РАЗРАБОТКА ПРОГРАММЫ. 18    1. Выбор средств программирования. 18    2. Разработка модулей 19    3. Разработка и описание функций 20 3. ТЕСТИРОВАНИЕ 23    1. Описание входных и выходных данных 23    2. Результаты тестирования 23   ЗАКЛЮЧЕНИЕ. 34  СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ… 35  ПРИЛОЖЕНИЕ А ТЕКСТ ПРОГРАММЫ  ПРИЛОЖЕНИЕ Б ГРАФИЧЕСКИЙ МАТЕРИАЛ | | | | | | | | | | |
|  |  |  |  |  | *КР.ПО4.**190344-01 81 00* | | | | | |
|  |  |  |  |  |
| *Изм* | *Лист* | докум № | П*одп.* | Д*ата* |
| *Разраб.* | | *Луд А.С.* |  |  | *Информационная система*  *«Книжный магазин». Пояснительная записка* | *Лит* | | | *Лист* | *Листов* |
| *Проверил* | | *Анфилец С.В.* |  |  |  | *К* |  | *3* | 35 |
|  | |  |  |  | *БрГТУ* | | | | |
| *Н. контр.* | | *Анфилец С.В.* |  |  |
| *Утв.* | |  |  |  |

Разработка и сравнительный анализ алгоритмов управления движением мобильного робота

по заданной траектории.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ВВЕДЕНИЕ**  Информация — основа современного общества. Объём её огромен и растёт с каждым годом. К настоящему времени человечеством накоплено гигантское количество информации об объектах и явлениях. Эта информация хранится в электронном виде и используется в базах данных.  База данных — это набор данных для информационных сетей и пользователей, осуществляющих хранение и обработку огромных информационных массивов.  База данных представляет собой определённым образом структурированную совокупность данных, хранящихся и обрабатывающихся в соответствии с некоторыми правилами. Как правило, база данных моделирует некоторую предметную область или её фрагмент. Очень часто в качестве постоянного хранилища информации баз данных выступают файлы.  Информация в базе данных может быть представлена как текст, изображение, таблица или объектно-ориентированная модель. Структурирование информации позволяет производить её анализ и обработку: делать пользовательские запросы, выборки, сортировки, вставки, производить математические и логические операции.  Преимущества баз данных:   1. Ёмкость хранения — базы данных хранят большое количество данных по сравнению   с другими хранилищами данных.   1. Несколько пользователей — разрешения могут быть предоставлены для   множественного доступа к базе данных. Это позволяет одновременно нескольким пользователям получить доступ и манипулировать данными.   1. Удаление данных — нежелательные требования нужно удалить из базы данных. В   таком случае, записи должны быть удалены из всех связанных таблиц, чтобы избежать  каких-либо нарушений данных.   1. Безопасность данных — эта особенность гарантирует, что злоумышленники не   получат доступ к данным, и что их качество поддерживается.  Система управления базами данных(СУБД) — это ПО, которое обеспечивает взаимодействие разных внешних программ с данными и дополнительные службы (журналирование, восстановление, резервное копирование и т.д.), в том числе посредством SQL. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **4** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Существует несколько видов СУБД по способу организации хранения данных:  Иерархические**.** Данные организованы в виде древовидной структуры. Пример — файловая система, которая начинается с корня диска и далее прирастает ветвями файлов разных типов и папок разной степени вложенности.  1)Сетевые**.** Видоизменение иерархической, у каждого узла может быть больше одного родителя.  2)Объектно**-**ориентированные**.** Данные организованы в виде классов/объектов c их атрибутами и принципами взаимодействия согласно ООП.  3)Реляционные**.** Данные этого вида СУБД организованы в таблицах. Таблицы могут быть связаны друг с другом, информация в них структурирована.  Для нашей информационной системы мы будем использовать объектно**-**ориентированный и реляционныйвид СУБД.  Внешние программы формируют запросы к СУБД на языке управления данными SQL. Одна из особенностей SQL – декларативность. То есть, SQL — декларативный язык. Это значит, что, вбивая команды, то есть, создавая запросы к SQL-серверу, мы описываем, что именно хотим получить, а не каким способом. Совершенно неважно, как и откуда сервер загрузит и сформирует интересующие нас данные. Главное – правильно сформулировать запрос.  Основные SQL-операторы подразделяют на следующие группы:  Data Definition Language (DDL) – определения данных. Создание структуры БД и её объектов;  Data Manipulation Language(DML) – собственно взаимодействие с данными: вставка, удаление, изменение и чтение;  Transaction Control Language (TCL) – управление транзакциями;  Data Control Language(DCL**)** – управление правами доступа к данным и структурам БД.  Для нашей информационной системы мы будем использовать все группы, особенно уделяя внимание DML**.**  JDBC  Для работы в Java с СУБД, для стандартизации работы с SQL-серверами, взаимодействие можно выполнять через единую точку — *JDBC* (Java DataBase Connectivity). Она представляет собой реализацию пакета *java.sql* для работы с СУБД. Производители всех популярных SQL-серверов выпускают для них драйверы JDBC. Используем экземпляры классов из *java.sql*. Затем мы передаем необходимые команды для получения/модификации данных. Далее *java.sql* через *jdbc-драйвер* взаимодействует с СУБД и возвращает нам готовый результат. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **5** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **1 ПОСТАНОВКА ЗАДАЧИ**  Необходимо разработать информационную систему «Книжный магазин», осуществляющую возможность взаимодействия пользователя с базой данных.  Предметная область: Книжный магазин.  Книжный магазин занимается продажей литературы разных жанров. В торговом зале книги располагаются на стеллажах, каждый из которых имеет свой номер. Любой клиент может зарегистрироваться и обратиться к продавцу-консультанту и получить информацию о выбранной им книге, и купить её.  Структура книжного магазина будет представлена в виде классов:  Классы: «Пользователь», «Книги».  «Пользователь»: фамилия: string  имя: string  отчество: string  дата рождения: int  телефон: int  адрес: string  логин: string  пароль: string    «Книги»: название: string  автор: string  стоимость: float  год издания: int  издание: string  количество страниц: int  статус: string | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **6** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| При проектировании ИС взгляды отдельных пользователей на предметную область называют локальными пользовательскими представлениями (ЛПП). Сведение этих взглядов в единую систему, выявление пересекающихся эпизодов и определение той части, которая необходима для решения поставленной задачи, разработчик ни в коей мере не может перекладывать на плечи пользователя. Этот этап является одним из основных при построении ИС. Его реализация невозможна без изучения таких процессов, которые протекают в изучаемой предметной области.  Завершение этапа приведет к формированию глобального пользовательского представления (ГПП), т.е. будет отражать точку зрения администратора БД. Курсовой проект был выполнен с целью практического освоения основных правил и приемов проектирования баз данных.  База данных была спроектирована в системе управления базами данных MySQL, а графический интерфейс пользователя — с помощью языка Java. В качестве предметной области разрабатываемой базы данных выбран «Книжный магазин”.  Реляционная база данных представляет собой множество взаимосвязанных таблиц, каждая из которых содержит информацию об объектах определенного вида. Каждая строка таблицы содержит данные об одном объекте (например, автомобиле, компьютере, клиенте), а столбцы таблицы содержат различные характеристики этих объектов - атрибуты (например, номер двигателя, марка процессора, телефоны фирм или клиентов).  Строки таблицы называются записями. Все записи таблицы имеют одинаковую структуру. Они состоят из полей (элементов данных), в которых хранятся атрибуты объекта. Каждое поле записи содержит одну характеристику объекта и представляет собой заданный тип данных (например, текстовая строка, число, дата). Для идентификации записей используется первичный ключ. Первичным ключом называется набор полей таблицы, комбинация значений которых однозначно определяет каждую запись в таблице.  Для курсового проекта необходимо выполнить следующие элементы:   1. Разработать реляционную базу данных в СУБД MySQL в соответствии с разработанной структурой; 2. Разработать графический интерфейс пользователя для созданной базы данных средствами языка программирования Java; 3. Реализовать функции чтения, добавления, поиска, удаления и сортировки базы данных через графический интерфейс пользователя; | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **7** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **1.1 Средства решения поставленной задачи**  MySQL — это реляционная система управления базами данных с открытым  исходным кодом. В настоящее время эта СУБД одна из наиболее популярных в веб-  приложениях — подавляющее большинство CMS использует именно MySQL (часто  только её, без альтернатив), а почти все веб-фреймворки поддерживают MySQL уже  на уровне базовой конфигурации (без дополнительных модулей).  Из преимуществ СУБД MySQL стоит отметить простоту использования, гибкость,  низкую стоимость владения (относительно платных СУБД), а также  масштабируемость и производительность.  MySQL позволяет хранить целочисленные значения со знаком и беззнаковые, длиной  в 1, 2, 3, 4 и 8 байтов, работает со строковыми и текстовыми данными фиксированной  и переменной длины, позволяет осуществлять SQL-команды SELECT, DELETE,  INSERT, REPLACE и UPDATE, обеспечивает полную поддержку операторов  и функций в SELECT- и WHERE- частях запросов, работает с GROUP BY и ORDER  BY, поддерживает групповые функции COUNT(), AVG(), STD(), SUM(), MAX()  и MIN(), позволяет использовать JOIN в запросах, в т.ч. LEFT OUTER JOIN и RIGHT  OUTER JOIN, поддерживает репликацию, транзакции, работу с внешними ключами  и каскадные изменения на их основе, а также обеспечивает многие другие  функциональные возможности.  Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов  таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие  полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции  на уровне отдельных записей. Есть и другие типы таблиц, разработанные  сообществом.  СУБД MySQL появилась в 1995. Написана на C и C++, протестирована на множестве  различных компиляторов и работает на различных платформах. C 2010 года  разработку и поддержку MySQL осуществляет корпорация Oracle. Продукт  распространяется как под GNU GPL, так и под собственной коммерческой лицензией.  Однако по условиям GPL, если какая-либо программа включает исходные коды  MySQL, то и эта программа тоже должна распространяться по лицензии GPL. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **8** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Java — объектно-ориентированный язык программирования, разработанный компанией SunMicrosystems (в последующем приобретенной компанией Oracle). Приложения Java обычно компилируются в специальный байт-код, поэтому они могут работать на любой виртуальной Java-машине (JVM) вне зависимости от компьютерной архитектуры. Программы на Java транслируются в байт-код, выполняемый виртуальной машиной Java (JVM) — программой, обрабатывающей байтовый код и передающей инструкции оборудованию как интерпретатор.  Достоинством подобного способа выполнения программ является полная независимость байт-кода от операционной системы и оборудования, что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Другой важной особенностью технологии Java является гибкая система безопасности благодаря тому, что исполнение программы полностью контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного доступа к данным или соединения с другим компьютером) вызывают немедленное прерывание.  Внутри Java существуют несколько основных семейств технологий:   * Java SE — JavaStandardEdition, основное издание Java, содержит компиляторы, API, JavaRuntimeEnvironment; подходит для создания пользовательских приложений, в первую очередь — для настольных систем; * Java EE — JavaEnterpriseEdition, представляет собой набор спецификаций для создания программного обеспечения уровня предприятия; * Java ME — JavaMicroEdition, создана для использования в устройствах, ограниченных по вычислительной мощности, например в мобильных телефонах, КПК, встроенных системах; * JavaFX — технология, являющаяся следующим шагом в эволюции Java как RichClientPlatform; предназначена для создания графических интерфейсов корпоративных приложений и бизнеса; | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **9** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1.2 Реализация базы данных**  Для реализации базы данных Книжного магазина было необходимо создать следующие таблицы:   1. Покупка – таблица, содержащая информацию о названии выбранных пользователем книг их цене и общей сумме за все выбранные книги; 2. Пользователь – таблица, содержащая в себе данные о пользователях. 3. Книги – таблица, содержащая в себе данные о книгах.   Атрибуты таблиц и типы данных, соответствующие значениям полей этих атрибутов для каждой из таблиц, указаны на схеме 1.   |  |  | | --- | --- | | Код | Числовой | | Фамилия | Текстовый | | Имя | Текстовый | | Отчество | Текстовый | | Дата рожд. | Числовой | | Телефон | Числовой | | Адрес | Текстовый | | Логин | Текстовый | | Пароль | Текстовый |   Покупка   |  |  | | --- | --- | | Код | Числовой | | Код\_продажи | Числовой | | Название | Текстовый | | Цена | Числовой | | Количество | Числовой | | Сумма | Числовой |     Пользователь     |  |  | | --- | --- | | Название | Текстовый | | Автор | Текстовый | | Стоимость | Числовой | | Год изд | Числовой | | Издание | Текстовый | | Кол. страниц | Числовой | | Статус | Текстовый |   Книги  Схема 1 – Описание атрибутов и типов данных  Данная схема полностью описывает типы значений атрибутов таблиц в соответствии с требованиями к базе данных. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **10** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Шифрование БД – это простейший способ защиты, при котором файл БД видоизменяется и становится недоступным для чтения с помощью стандартных служебных программ или текстовых редакторов. Шифрование незащищенной БД неэффективно, поскольку исходную БД можно открыть и получить полный доступ к ее объектам. Шифрование применяется при электронной передаче БД или сохранении ее на внешних носителях.  Использование специальных параметров запуска позволяет задать стартовую форму, которая автоматически открывается при открытии БД. При этом можно скрыть окно БД и установить собственную кнопочную форму. Пользователь может выполнять с БД действия, которые допускает интерфейс.  Простым способом защиты является установка пароля для открытия БД. При  каждом открытии БД будет появляться диалоговое окно, в которое требуется ввести  пароль. Может быть установлен единый пароль для всех пользователей, но наиболее  гибким является способ, при котором пароль присваивается каждому пользователю.  Для каждого пользователя могут быть определены объекты, доступ к которым он получает.  Пользователь, создавший таблицу, является ее владельцем. Это означает, что пользователь имеет все привилегии в созданной им таблице и может передавать привилегии в этой таблице другим пользователям.  Для пользователя таблицы могут быть назначены следующие привилегии:   * разрешение выполнять запросы в таблице: * разрешение выполнять добавление новой строки в таблице; * разрешение выполнять изменение значений полей в таблице; * разрешение выполнять удаление записей в таблице; * разрешение выполнять сортировку столбцов в таблице. * Разрешение выполнять поиск записей по заданному значению. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **11** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| По созданной логической структуре была разработана база данных средствами СУБД MySQL Server. В неё входят 3 таблицы: book, users, sales, product. В свою очередь таблицы содержат столбцы.  В таблицу «book» входят поля: «id» типа int(11), «bname» типа varchar (255), «author» типа varchar (255), «price» типа float, «year» типа int(11), «izdatel» типа varchar (255), «numstr» типа int(11), «status» типа varchar (255).  В таблицу «users» – «idk» типа int(11), «fname» типа varchar (255), «lname» типа varchar (255), «sname» типа varchar (255), «born» типа int(11), «phone» типа int (11), «adress» типа varchar (255), «login» типа varchar (255), «password» типа varchar (255).  Столбцы всех первичных ключей таблицы являются идентифицирующими, т.е. значения этих полей генерируются автоматически.  Структуры БД:  Таблица **book**    Таблица **users** | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **12** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| В таблицу «sales» входят поля: «id» типа int(11), «subtotal» типа int (11), «pay» типа int (11), «balance» типа int(11).  В таблицу «product» – «id» типа int(11), «sales\_id» типа int(11), «bname» типа varchar (255), «price» типа int (11), «count» типа int(11), «total» типа int (11).  Столбцы всех первичных ключей таблицы являются идентифицирующими, т.е. значения этих полей генерируются автоматически.  Структуры БД:  Таблица **product**    Таблица **sales** | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **13** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **1.3 Запуск серверной части**  В комплекте OpenServer существует целый набор инструментов, обладающих графическим интерфейсом. Поэтому, если вам нужно будет создать базу MySQL в OpenServer или поменять пароль пользователя, то использование графических редакторов — это самый лёгкий и простой способ сделать задуманное.  Визуальные инструменты представлены различными утилитами, доступ к которым можно получить из меню OpenServer, при выборе пункта  **Дополнительные(**в нашем случае выбираем phpMyAdmin (см. рисунок 1**))**:    Рисунок 1 – запуск phpMyAdmin с помощью OpenServer  phpMyAdmin – универсальный и достаточно мощный инструмент для управления базами данных (БД) для разработчиков, администраторов, а также для обычных пользователей — владельцев сайтов. Предоставляемый phpMyAdmin веб-интерфейс позволяет получать доступ к управлению БД из любого браузера в любой операционной системе (ОС). Поэтому когда возникает необходимость в выборе клиента MySQL, то phpMyAdmin – это пожалуй самое оптимальное решение, в особенности для веб-разработчиков и обычных пользователей.  После запуска из меню OpenServer phpMyAdmin запустится либо в открытом окне вашего браузера, либо в отдельном окне(см рисунок 2): | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **14** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Рисунок 2 – отображение phpMyAdmin в окне браузера  Это стартовый экран, на котором нужно ввести имя пользователя MySQL и его пароль для подключения к серверу. По умолчанию **root/root**(см. рисунок 3)**:**    Рисунок 3 – главное окно интерфейса phpMyAdmin  Перейдем на вкладку **Базы данных**. Под меткой **Создать базу данных** введем какое-нибудь имя для новой бд, например, **bookshop** и нажмем на кнопку "Создать".  Новая база данных пока пуста и не содержит ничего. Добавим в нее таблицу, которая будет хранить данные. Для этого нажмем на название базы данных и мы попадем на вкладку "Структура", где нам будут предложены опции новой таблицы. В поле "Имя" введем название новой таблицы. Пусть, таблицы будет хранить данные о книгах, поэтому введем название "book", а в качестве количества столбцов введем цифру 8.  Для создания таблицы нажмем на кнопку "Вперед". После этого у нас появится набор ячеек для установки параметров столбцов. Укажем последовательно для имен столбцов следующие: id, bname, … . В качестве типа укажем для столбцов id тип **INT**, а для столбца bname - тип **VARCHAR**. Для столбца name в поле "Длина/Значения" укажем число 255 - оно будет указывать максимальную длину строки в символах. Также для столбца id укажем в поле "Индекс" **PRIMARY** а в поле "AI" (AutoIncrement) поставим галочку. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **15** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1. **РАЗРАБОТКА АЛГОРИТМОВ**   **2.1 Алгоритмы общего задания**  **Алгоритм чтения данных из базы данных**  Исходные данные: Таблица SQL, UI  Алгоритм:  1. Получение данных из таблицы SQL.  2. Передача данных в UI.  **Алгоритм добавления новых элементов**  Исходные данные: Таблица SQL и входные данные.  Алгоритм:  1. Ввод новых данных.  2. Получение данных.  3. Добавление полученных данных в таблицу SQL.  4. Вывод данных в UI.  **Алгоритм редактирования записи в базе данных**  Исходные данные: Таблица SQL, id обновляемой записи  Алгоритм:  1. Найти запись в таблице по id.   1. Выделить нажатием ЛКМ желаемую запись. 2. Заменить текущие значения полей записи на введённые данные. 3. Вывод данных с изменением в UI. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **16** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Алгоритм удаления выбранного элемента**  Исходные данные: Таблица SQL, id удаляемой записи  Алгоритм:   1. Найти запись в таблице по id. 2. Выделить нажатием ЛКМ желаемую запись. 3. Удалить данную запись. 4. Вывод данных с удалением в UI.   **Поиск по записям в базе данных**  Исходные данные: Таблица SQL и её данные  Алгоритм:  1. Введение желаемых данных для поиска.  1. Получение данных о записи.  3. Вывод найденных в UI.  **Сортировка по записям в базе данных**  Исходные данные: Таблица SQL и её данные  Алгоритм:   1. Ввод в коде программы столбца для сортировки.   2. Нажатие во время выполнения программы кнопки «Сортировать».  3. Выбор типа сортировки.  4. Нажатие кнопки «ок».  5. Вывод отсортированных данных в БД и UI. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **17** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1. **РАЗРАБОТКА ПРОГРАММЫ**    1. **Выбор средств программирования**   Среда разработки: NetBeans IDE 8.2. ОС: Windows 10.  В ходе разработки приложения базы данных Книжного магазина был использован язык Java как средство разработки графического интерфейса пользователя.  Для реализации использовалась библиотека Swing, являющийся частью библиотеки JFC (JavaFoundationClasses), представляющей собой набор библиотек для создания графических оболочек. Компоненты Swing разработаны для одинаковой кроссплатформенной работы.  Начиная с версии Java 1.2, датированной 1998 годом, Swing входит в состав JavaRuntimeEnvironment.  JavaRuntimeEnvironment (JRE) — минимальная реализация виртуальной машины, необходимая для исполнения Java-приложений, без компилятора и других средств разработки. Состоит из виртуальной машины и библиотеки Java-классов. JRE распространяется компанией Oracle свободно.  Главное окно приложения(меню)  Вкладка  Книги  Вкладка  Покупка  Вкладка  Пользователи  Таблица  Покупка  Таблица  Пользователи  Таблица  Книги  Схема 1 – Структура переходов между вкладками в программном проекте  Схема отражает переходы между вкладками главного окна графической оболочки. При открытии вкладки открывается таблица базы данных, соответствующая этой таблице, и дальнейшая работа производится только с этой таблицей до момента, пока пользователь не вернётся в главное меню и не переключит вкладку. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **18** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **3.2 Разработка модулей**  Программа разбита на 6 модуля: book.java, klient.java, login.java, User.java, buybook.java, Main.java.  При разработке программы были использованы следующие типы данных: целочисленный тип int, строковый тип String, тип с плавающей точкой float, тип без значения void.  **Модуль book.java**  Содержит окно с таблицей БД о книгах и операциями над их данными.  **Модуль klient.java**  Содержит окно с таблицей БД о пользователях и операциями над их данными.  **Модуль login.java**  Содержит окно с текстовыми полями для авторизации пользователей.  **Модуль User.java**  Содержит основной класс **User\User1** конструктор **User\User1** и методы **get**.  **Модуль buybook.java**  Содержит окно для реализации покупки пользователем выбранных книг.  **Модуль Main.java**  Содержит окно для организации меню. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **19** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **3.3 Разработка и описание функций**    **Connect()** – функция для подключения к БД MySQL.  Для того, чтобы подключить базу данных MySQL, нам потребуется четыре вещи:   1. Строка подключения JDBC (например: jdbc:mysql://localhost:3306/book). 2. Имя пользователя (root). 3. Пароль (root). 4. База данных с некоторым количеством таблиц для примера (например, база данных книг).   Строка подключения для MySQL начинается с jdbc:mysql. Это название протокола соединения, за которым следуют хост и порт подключения, на которых запущена база данных. В нашем случае это localhost с портом по умолчанию 3306 (если вы его не поменяли при установке). Следующая часть — book— имя базы данных, которая уже существует в MySQL.  Для базы данных MySQL мы будем использовать драйвер Type 4 JDBC из пакета mysql-connector-java-5.1.49-bin.jar. Он написан на чистой Java, а значит, нам не понадобятся какие-либо нативные библиотеки или ODBC-мост. Все, что нам надо будет сделать — это положить JAR-файл в директорию, содержащуюся в CLASSPATH. JAR-файл содержит класс com.mysql.jdbc.Driver, необходимый для подключения к MySQL.    import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  Connection conn;  PreparedStatement pst;  ResultSet rs;  public void Connect(){  try {  Class.forName("com.mysql.jdbc.Driver");  } catch (ClassNotFoundException ex) {  Logger.getLogger(book.class.getName()).log(Level.SEVERE, null, ex);  }  try {  conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/bookshop", "root", "root");  } catch (SQLException ex) {  Logger.getLogger(book.class.getName()).log(Level.SEVERE, null, ex);  }  } | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **20** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **import java.sql.SQLException;** - обработчик ошибок  Вызов методов нашего кода может вернуть ошибки, на которые следует обратить внимание. Ошибки при работе с СУБД — это чаще всего **SQLException**.    catch (SQLException ex) {  Logger.getLogger(book.class.getName()).log(Level.SEVERE, null, ex);  }  **Class.forName**("com.mysql.jdbc.Driver") – убеждаемся в наличии соответствующего JDBC-драйвера (который мы ранее загрузили и установили).  **DriverManager.getConnection**("jdbc:mysql://localhost:3306/bookshop", "root", "root") – устанавливаем соединение СУБД. По переданному адресу, JDBC сама определит тип и местоположение нашей СУБД и вернёт Connection, который мы можем использовать для связи с БД.  **book\_load() –** вывод таблицы и её обновлений при изменении данных в UI  import java.util.Vector;  public void book\_load(){  try {  pst = conn.prepareStatement("select \* from book");  rs = pst.executeQuery();  ResultSetMetaData rsd = rs.getMetaData();  int c;  c = rsd.getColumnCount();  DefaultTableModel d = (DefaultTableModel)jTable1.getModel();  d.setRowCount(0);  while(rs.next()){  Vector v2 = new Vector();  for(int i=1; i<=c; i++){  v2.add(rs.getString("id"));  v2.add(rs.getString("bname"));  v2.add(rs.getString("author"));  v2.add(rs.getString("price"));  v2.add(rs.getString("year"));  v2.add(rs.getString("izdatel"));  v2.add(rs.getString("numstr"));  v2.add(rs.getString("status"));  }  d.addRow(v2);  }  } catch (SQLException ex) {  Logger.getLogger(book.class.getName()).log(Level.SEVERE, null, ex);  }  } | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **21** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Класс Vector** реализует динамический массив. Он похож нa ArrayList , но с двумя отличиями:  1)Vector синхронизирован.  2)Vector содержит много устаревших методов, которые не являются частью структуры коллекций.  В Java класс Vector оказывается очень полезным, если вы заранее не знаете размер массива или вам нужен только тот, который может изменять размеры за время жизни программы.  **executeQuery()** – для чтения данных из СУБД.  **executeUpdate()** – для модификации данных в СУБД.  Непосредственно ответ сервера в виде **ResultSet** можно обработать, итерируя его через **first()**, **last()**, **next()** и так далее. Отдельные поля результаты мы можем получить через геттеры: **getInteger(), getString().**  Для получения данных из БД вы можете выполнить SELECT-запрос.  pst = conn.prepareStatement("select \* from book");  rs = pst.executeQuery();  query = "SELECT \* FROM book ORDER BY author ASC";  query = "SELECT \* FROM book ORDER BY numstr DESC";  st = conn.createStatement();  rs = pst.executeQuery(query);  Добавление данных мало отличается от их получения: мы просто используем INSERT-запрос вместо SELECT-запроса и метод executeUpdate() вместо executeQuery(). Этот метод используется для запросов INSERT, UPDATE и DELETE.  pst = conn.prepareStatement("update book set bname = ?, author = ?, price = ?, year = ?, izdatel = ?, numstr = ?, status = ? where id = ?");  int k = pst.executeUpdate();  pst = conn.prepareStatement("insert into book(bname, author, price, year, izdatel, numstr, status)values(?,?,?,?,?,?,?)");  int k = pst.executeUpdate();  pst = conn.prepareStatement("delete from book where id = ?");  int k = pst.executeUpdate();  **sales() –** получение данных из таблицы о книгах, реализация покупки выбранных книг в выбранном количестве с подсчётом общей суммы и сдачи при покупке книги. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **22** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1. **ТЕСТИРОВАНИЕ**    1. **Описание входных и выходных данных**   Формами входных данных является табличный компонент, т.к. он позволяет редактировать все свои ячейки, кроме тех, которые отвечают за отображение первичного ключа, который, по сути, является счетчиком.  Формой выходных данных также являются табличные компоненты, т.к. после каких-либо изменений в таблице базе данных соответствующий ей табличный компонент отображает произведенные изменения.   * 1. **Результаты тестирования**   Среда тестирования – ПК, процессор Intel Core i5-7200U с частотой 2.5 - 2.71 ГГц, ОЗУ 4 ГБ, тип системы: 64-разрядная OC Windows 10 Домашняя. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **23** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Тест 1:** «Авторизация пользователя»  Ожидаемый результат: регистрация нового пользователя в бд или вход.  Описание: пользователь вводит логин и пароль. Если он уже является пользователем то все выполнится, но если он введет неправильные данные или вообще ничего не введёт, то UI оповестит об этом (см. рисунки 4.1.1 — 4.1.9).  Полученный результат:    Рисунок 4.1.1 – главный экран авторизации Рисунок 4.1.2 – нажатие «Войти» без ввода данных    Рисунок 4.1.3 – ввод неверных данных Рисунок 4.1.4 – нажатие «Войти» при неверных данных | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **24** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Рисунок 4.1.5 – главный экран «Зарегистрироваться»      Рисунок 4.1.6 – добавлены 3 новых пользователя | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **25** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Рисунок 4.1.7- ввод данных одного из пользователей Рисунок 4.1.8 – нажатие «Войти» при верных данных    Рисунок 4.1.9 – авторизация 3 пользователей.    Вывод: авторизация пользователей выполнена успешно. Ожидаемый результат совпал с полученным. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **26** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Тест 2:** «Добавление новых книг»  Ожидаемый результат: добавление новых книг в бд и в UI.  Описание: пользователь вводит новые данные о книге и нажатием клавиши «Добавить» добавляет запись(см. рисунок 4.2).  Полученный результат:      Рисунок 4.2 – добавление 4 записей  Вывод: добавление новых книг выполнено успешно. Ожидаемый результат совпал с полученным. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **27** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Тест 3:** «Редактирование и обновление данных о книгах»  Ожидаемый результат: изменение данных, введённых пользователем.  Описание: пользователь нажатием ЛКМ выбирает запись и редактирует её данные, обновление происходит автоматически (см. рисунок 4.3).  Полученный результат:        Рисунок 4.3 – изменяем поля «Цена» для 1 записи на **87.44р** , «Год издания» для 2 записи на **2008**, «Издательство» для 2 записи на **Анатомия**, «Статус» для записи 4 на **Выбрано**.  Вывод: редактирование данных выполнено успешно. Ожидаемый результат совпал с полученным. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **28** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Тест 4:** «Удаление записи»  Ожидаемый результат: удаление записи выбранной пользователем.  Описание: пользователь нажатием ЛКМ выбирает запись и удаляет её (см. рисунок 4.4).  Полученный результат:        Рисунок 4.4 – удаляем запись 3, **Геометрия**  Вывод: удаление выбранной записи работает корректно. Ожидаемый результат совпал с полученным. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **29** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Тест 5:** «Сортировка данных выбранного столбца»  Ожидаемый результат: данные выбранного столбца будут отсортированы.  Описание: Пользователь нажатием кнопки «Сортировать» открывает окно с выбором типа сортировки(по возрастанию\убыванию) и выбрав тип сортирует данные(см. рисунок 4.5).  Полученный результат:      Сортировка( по возрастанию) столбца «Автор»    Сортировка( по убыванию) столбца «Кол. страниц»    Рисунок 4.5 – сортировка данных  Вывод: сортировка выбранного столбца работает корректно. Ожидаемый результат совпал с полученным. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **30** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Тест 6:** «Поиск записи»  Ожидаемый результат: будет найдена запись по введённым данным.  Описание: Пользователь вводит в текстовое поле данные по которым хочет найти всю информацию об записи(см. рисунок 4.6).  Полученный результат:              Рисунок 4.6 – найденные данные  Вывод: поиск по введённым данным работает корректно. Ожидаемый результат совпал с полученным. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **31** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Тест 7:** «Покупка книг» | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **32** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **33** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ЗАКЛЮЧЕНИЕ**  Использование информационных систем в нашем мире имеет глобальный характер.  Информационные системы дают возможность систематизировать информационные ресурсы общества. Исследования показывают, что грамотное пользование информационных ресурсов значительно экономит природные ресурсы нашей планеты.  Базы данных позволяют оптимизировать и, в первую очередь, автоматизировать информационные процессы. Во многих развитых странах основная часть работоспособного населения принимает участие в обработке, хранении и передаче информационных продуктов и услуг.  Использование БД обеспечивает надежное хранение большого количества данных и легкое манипулирование ими. При этом БД обеспечивают экономные затраты памяти компьютерных систем.  Все поставленные задачи были выполнены успешно, а условия соблюдены. Были разработаны алгоритмы для решения поставленных задач, проведено тестирование правильности выполнения программы. Все требования работают корректно, а ожидаемый результат совпал с полученным.  В процессе выполнения курсового проекта была исследована предметная область Книжного магазина, реализованы этапы проектирования базы данных, построены информационно-логическая и логическая схемы данных, составлена структура базы данных, соответствующая СУБД MySQL.  В ходе разработки курсового проекта были изучены такие языки, как SQL и Java, а также изучена методика проектирования баз данных и их создание в СУБД MySQL.  В результате выполнения курсового проекта было получено приложение базы данных Книжного магазина в виде настольного приложения на языке Java. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **34** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**   1. ГОСТ 2.105-95. Единая система конструкторской документации (ЕСКД). Общие требования к текстовым документам. 2. ГОСТ 19.504-79. Единая система программной документации ЕСПД. Руководство программиста. Требования к содержанию и оформлению. 3. ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения.   4. ГОСТ 19.005-85. ЕСПД.  Р-схемы алгоритмов и программ. Обозначения условные графические и правила выполнения.   1. ГОСТ 19.101-77. ЕСПД. Виды программ и программных документов. 2. ГОСТ 19.102-77. ЕСПД. Стадии разработки. 3. ГОСТ 19.103-77. ЕСПД. Обозначения программ и программных документов. 4. ГОСТ 19.401-78. ЕСПД. Текст программы. Требования к содержанию и оформлению. 5. ГОСТ 19.402-78. ЕСПД. Описание программы. 6. Шилдт Г. Java. Полное руководство, 8-е изд. – 2012. 7. Мейер М. Теория реляционных баз данных. – 2006. 8. ГОСТ 7.1-2003. Система стандартов по информации, библиотечному и   издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **35** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |

