Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Локтионова Оксана Геннадьевна
Должность: проректор по учебной работе
Дата подписания: 05.04.2023 14:16:33
Уникальный программный ключ:
0b817ca911e6668abb13a5d426d39e5f1c11eabbf73e943df4a4851fda56d089

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра информационной безопасности

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«_31_» _03_ 2023 г.

ПРОГРАММНО-АППАРАТНАЯ ЗАЩИТА ИСПОЛНЯЕМОГО МОДУЛЯ ОТ НЕСАНКЦИАНИРОВАННОГО ДОСТУПА

Методические указания по выполнению практических работ для студентов укрупненной группы специальностей и направлений подготовки 10.00.00

Курск 2023

УДК 004.424

Составители: А.В. Митрофанов, А.А. Чеснокова

Рецензент

Кандидат технических наук, доцент кафедры «Информационная безопасность» А.Л. Марухленко

Программно-аппаратная защита исполняемого модуля от несанкционированного доступа: методические указания по выполнению практической работы по дисциплине «Технологии и методы программирования» / Юго-Зап. гос. ун-т; сост.: А.В. Митрофанов, А.А. Чеснокова. Курск, 2023. 11 с., Библиогр.: с. 11.

Содержат основные теоретические и практические сведения о программно-аппаратной защите исполняемого модуля от несанкционированного доступа. Указывается порядок выполнения практической работы, правила оформления и содержание отчета.

Методические указания по выполнению практических работ по дисциплине «Технологии и методы программирования», предназначены для студентов укрупненной группы специальностей и направлений подготовки 10.00.00

Текст печатается в авторской редакции

Подписано в печать _____. Формат 60×84 1/16.
Усл.печ.л. . Уч.-изд.л. . Тираж 50 экз. Заказ 218. Бесплатно
Юго–Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

СОДЕРЖАНИЕ

1. ЦЕЛЬ РАБОТЫ

Цель практической работы – предложить программно-аппаратную защиту исполняемого модуля от несанкционированного доступа.

2. ЗАДАНИЕ

Ознакомиться с теоретическим материалом. Ознакомиться с примерами решения. Предложить программно-аппаратную защиту исполняемого модуля от несанкционированного доступа.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Получить задание.
2. Изучить теоретическую часть.
3. Предложить программно-аппаратную защиту исполняемого модуля от несанкционированного доступа.

4. СОДЕРЖАНИЕ ОТЧЕТА

1. Титульный лист.
2. Краткая теория.
3. Предложения по программно-аппаратной защите исполняемого модуля от несанкционированного доступа.
4. Вывод.

5. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Программно-аппаратная защита используется для защиты программного обеспечения от несанкционированного (неавторизованного) доступа и нелегального использования. Защитный механизм программным образом опрашивает специальное устройство, используемое в качестве ключа, и работает только в его присутствии. Таким образом, механизм программно-аппаратной защиты содержит две составляющие:

1) аппаратное устройство (аппаратная часть);

2) программный модуль (программная часть).

Поэтому обычно говорят о системах программно-аппаратной защиты. Очевидно, что стоимость такого механизма превышает стоимость программной защиты, причем стоимость аппаратной части, как правило, превышает стоимость программной части. По этой причине программно-аппаратная защита считается привилегией корпоративных заказчиков, так как для индивидуального пользователя часто неприемлема с экономической точки зрения. Обратим внимание на то, что по существу программно-аппаратная защита не является защитой программ от нелегального распространения и использования. Не станет заказчик программы оплачивать дорогую аппаратуру только ради соблюдения авторских прав разработчика. Но если программный продукт снабжен модулем, предназначенным для защиты от несанкционированного доступа к данным и информации пользователя, то заказчик, как правило, готов платить за

аппаратуру, повышающую надежность такой защиты. Система защиты от несанкционированного доступа к данным реализована таким образом, что осуществляет проверку легальности пользователя при работе с программным обеспечением и тем самым косвенно препятствует и незаконному использованию программы. Кроме того, современные аппаратные устройства (ключи), помимо информации о законном пользователе, могут содержать также информацию о программном продукте. А системы программно-аппаратной защиты, кроме аутентификации пользователя, могут производить аутентификацию приложения. Поэтому системы программно-аппаратной защиты от несанкционированного доступа могут служить в то же время и для защиты авторских прав разработчиков программ. Системы программно-аппаратной защиты широко используются на практике и многими пользователями признаются надежным средством.

Защита от несанкционированного доступа Рассмотрим основные моменты защиты информации от несанкционированного доступа. Речь идет о таком порядке работы, при котором

1) доступ к информации имеет только тот пользователь, который имеет разрешение; будем называть такого пользователя законным;

2) каждый законный пользователь работает только со своей информацией и не имеет доступа к информации другого законного пользователя;

3) каждый законный пользователь может выполнять только те операции, которые ему разрешено выполнять.

Для организации такого порядка работы прежде всего необходимо обеспечить распознавание законного пользователя. Этот процесс часто называют авторизацией пользователя. Авторизация пользователя включает три этапа.

1. Идентификация пользователя.

2. Аутентификация пользователя.

3. Непосредственно авторизация пользователя.

Идентификация пользователя (identification) - это, с одной стороны, присвоение пользователю идентификатора - некоторого уникального признака (или нескольких); с другой стороны, процесс, во время которого пользователь указывает присвоенный ему идентификатор. Другими словами, идентификация - это процесс, при котором пользователь называет себя. Аутентификация пользователя (от англ. autentication - установление подлинности) - установление подлинности пользователя на основе сравнения с эталонным идентификатором. Авторизация пользователя - установление прав пользователя. Авторизованный пользователь (авторизованное лицо) - пользователь (лицо), который получил определенные права на работу с информацией. В процессе авторизации для законного пользователя определяются права пользователя, то есть определяются данные, с которыми ему разрешено работать; операции, которые ему разрешено выполнять и т.п

6. ПРИМЕР ВЫПОЛНЕНИЯ РАБОТЫ

Правила связи программных модулей по информации

1. Информация зон глобальных переменных доступна дня использования любым модулем, входящим и комплекс программ или группу программ в соответствии с областью действия зоны глобальных переменных, т.е. глобальные переменные, могут быть доступны не для всего комплекса программ, а лишь для указанной в описании группы модулей.

2. Локальные переменные доступны лишь в пределах того модуля, в котором они определены или объявлены.

3. Для взаимодействия вызываемых и вызывающих модулей создаются зоны обменных переменных, информация из которых доступна лишь модулям, непосредственно связанным по управлению.

4. После окончания работы вызываемого модуля считается, что регистры не содержат информации, являющийся результатом его работы. Запрещается их использовать в вызывающем модуле.

5. Информация, находящаяся в регистрах вызывающего модуля, при вызове должна быть сохранена на период выполнения вызываемого модуля и восстановлена при возврате управления в вызывающий модуль. Сохранение регистров может осуществлять как вызывающий, так и вызываемый модуль.

Правила связи программных модулей по управлению

1. Передача управления вызываемому модулю всегда осуществляется через его начало, т.е. через первый оператор или команду.

2. Если необходимо исполнить модуль с некоторой внутренней точки, то вызов осуществляется стандартным образом (через первый оператор), а точка начала задается в виде параметра, при этом в начале вызываемого модуля должен стоять переключатель, который обеспечивает передачу управления к внутренним точкам входов по параметру, указанному при обращении.

3. Выход из вызываемого модуля всегда происходит через его естественное окончание, т.е. после нормального его завершения.

4. По окончании исполнения вызываемого модуля управление передастся в вызывающий модуль на оператор, следующий непосредственно за оператором вызова.

5. Модули низших уровней или одного уровня иерархии могут

вызываться для исполнения только моделями высших уровней, т.е. модули низших уровней не могут вызывать модули высших уровней, а модули одного уровня - вызывать друг друга.

6. В каждом модуле должна быть предусмотрена возможность подключения контрольных и отладочных средств; операторы, реализующие эти средства, обычно сосредоточиваются в конце модуля.

7. СПИСОК ВОПРОСОВ

1. Что такое программный модуль?
2. Что такое защищенное программирование?
3. Что такое объектный модуль?
4. Что такое загрузочный модуль?
5. Правила связи программных модулей по управлению.
6. Правила связи программных модулей по информации.

8. БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Дж.Хьюз, Дж.Мичтом. Структурный подход к программированию. М.: Мир, 1980. - С. 29-71.

2. В.Турский. Методология программирования. - М.: Мир, 1981. - С. 90-164.

3. Е.А.Жоголев. Технологические основы модульного программирования//Программирование,1980, #2. - С. 44-49.

4. R.C.Holt. Structure of Computer Programs: A Survey//Proceedings of the IEEE, 1975, 63(6). - P. 879-893.

5. Г.Майерс. Надежность программного обеспечения. М.: Мир, 1980. - С. 92-113.

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра информационной безопасности

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«_31_» _03_____ 2023 г.

РАЗРАБОТКА ПРОЕКТА ДЛЯ ВЗАИМОДЕЙСТВИЯ С СЕРВЕРОМ В ПРОГРАММЕ POSTMAN

Методические указания по выполнению практических работ для студентов укрупненной группы специальностей и направлений подготовки 10.00.00

Курск 2023

УДК 004.424

Составители: А.В. Митрофанов, А.А. Чеснокова

Рецензент

Кандидат технических наук, доцент кафедры «Информационная безопасность» А.Л. Марухленко

Разработка проекта для взаимодействия с сервером в программе Postman: методические указания по выполнению практической работы по дисциплине «Технологии и методы программирования» / Юго-Зап. гос. ун-т; сост.: А.В. Митрофанов, А.А. Чеснокова. Курск, 2023. 12 с., Библиогр.: с. 12.

Содержат основные теоретические и практические сведения о разработке проекта для взаимодействия с сервером в программе Postman. Указывается порядок выполнения практической работы, правила оформления и содержание отчета.

Методические указания по выполнению практических работ по дисциплине «Технологии и методы программирования», предназначены для студентов укрупненной группы специальностей и направлений подготовки 10.00.00

Текст печатается в авторской редакции

Подписано в печать _____. Формат 60×84 1/16.
Усл.печ.л. . Уч.-изд.л. . Тираж 50 экз. Заказ 219. Бесплатно
Юго–Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

СОДЕРЖАНИЕ

1. ЦЕЛЬ РАБОТЫ

Цель практической работы – разработать проект для взаимодействия с сервером в программе Postman.

2. ЗАДАНИЕ

Ознакомиться с теоретическим материалом. Ознакомиться с примерами решения. Разработать проект для взаимодействия с сервером в программе Postman.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Получить задание.
2. Изучить теоретическую часть.
3. Разработать проект для взаимодействия с сервером в программе Postman.

4. СОДЕРЖАНИЕ ОТЧЕТА

1. Титульный лист.
2. Краткая теория.
3. Основные этапы работы.
4. Вывод.

5. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Программа Postman предназначена для тестирования работы API, а также для отправки запросов POST и GET. В отличие от похожей утилиты curl, она имеет графический интерфейс, поэтому легко осваивается даже новичками.

Главное окно программы разделено на четыре области. Разделение на блоки идет по функционалу, что заметно упрощает настройку и управление. (рис.1)

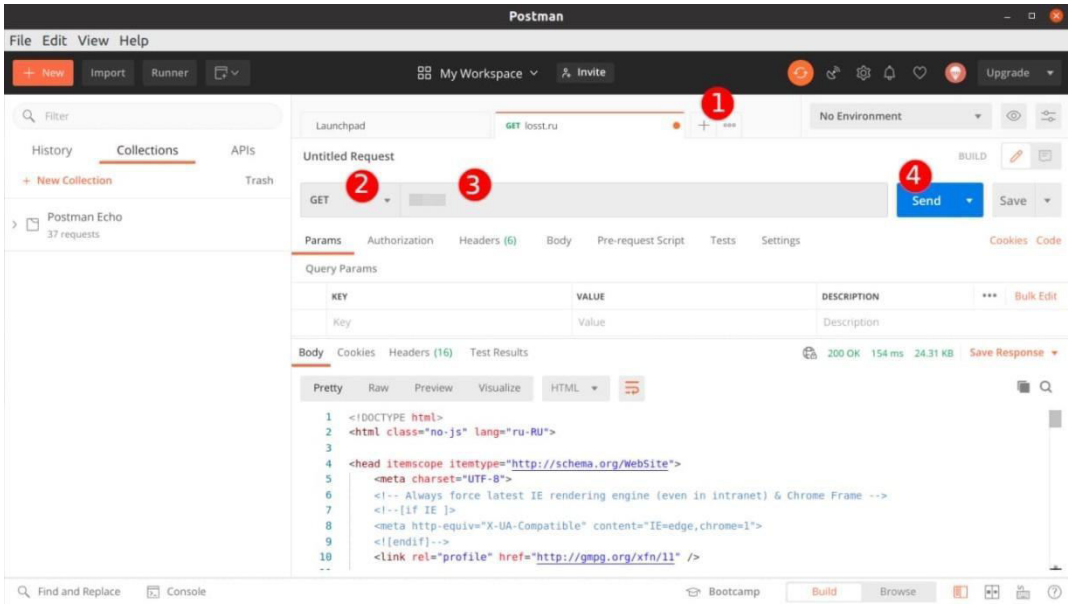

Рисунок 1 – окно программы Postman

Описание меню

1. Верхняя панель – здесь расположены основные настройки программы.

2. Боковая панель – сюда выделены запросы, выполненные ранее или сохраненные в качестве «избранного».

3. Панель вкладок – инструмент переключения между разными запросами.

4. Рабочая область – все базовые настройки отправленного запроса, перечень возвращаемых по нему данных.

Метод GET используется для запроса URL-адреса с веб-сервера для получения документов HTML. Для браузеров это обычный метод доставки информации, которая считается частью протокола HTTP. Метод GET представлен в виде URL, так что его можно добавить в закладки. GET широко используется в поисковых системах. После отправки запроса пользователем в поисковую систему, движок выполняет запрос и выдает полученную страницу. Результаты запроса могут быть установлены в виде ссылки (в закладки).

Метод GET позволяет генерировать якоря, что помогает в доступе к программе CGI с запросом, исключающим использование формы. Запрос состоит из ссылки, поэтому при посещении ссылки программа CGI извлекает подходящую информацию из базы данных.

У метода GET есть некоторые проблемы с безопасностью, потому что вставленные данные видны в URL. Только ограниченный объем данных может быть передан через метод GET, так как длина URL-адреса, которую браузер может просматривать, может составлять тысячу символов.

Другая проблема, связанная с методом GET, заключается в том, что он не может работать с иностранными языками. Метод

GET не рекомендуется использовать, но все же, когда атрибуты метода не определены, метод GET используется по умолчанию.

Метод POST подходит в условиях, когда может пройти значительное количество информации. Когда сервер получает запрос с помощью формы, использующей POST, он продолжает «прослушивать» оставшуюся информацию. Проще говоря, метод передает всю релевантную информацию, введенную в форму, сразу после выполнения запроса к URL.

Метод POST должен установить два контакта с веб-сервером, а GET только один. Запросы в POST обрабатываются так же, как и в методе GET, где пробелы представлены знаком плюс (+), а остальные символы закодированы в шаблоне URL. Он также может отправлять элементы файла.

Ключевые различия между GET и POST методом

1. Метод GET помещает параметры в URI, а метод POST добавляет параметры в тело.

2. GET в основном используется для получения информации. В отличие от этого, целью метода POST является обновление данных.

3. Результаты запроса POST не могут быть добавлены в закладки, тогда как результаты запроса GET могут быть добавлены в закладки, потому что они существуют в форме URL.

4. В методе GET информация отображается в URL, что увеличивает уязвимости и риск взлома. В отличие от этого, метод POST не отображает переменную в URL, и в нем также можно

использовать несколько методов кодирования, что делает его устойчивым.

5. Когда в форме используется метод GET, в типах данных принимаются только символы ASCII. Наоборот, метод POST не связывает типы данных форм и допускает двоичные, а также символы ASCII.

6. Размер переменной в методе GET составляет около 2000 символов. И наоборот, метод POST допускает переменный размер до 8 Мб.

7. Данные метода GET кэшируются, а данные метода POST нет.

6. ПРИМЕР ВЫПОЛНЕНИЯ РАБОТЫ

Выполнение запроса

Выполнение простого запроса, без сохранения в коллекции, осуществляется кликом по кнопке со значком . В результате откроется новая вкладка, где есть возможность выбрать тип запроса (GET или POST) и внести домен, который будет открываться. Остается нажать на кнопку Send, которая и запустит процедуру проверки.

В нижней части страницы появится код страницы (HTML). Здесь имеется несколько вкладок:

1. Body – данные, содержащиеся в теле запроса.

2. Cookie – информация, записанная сервером.

3 Headers – заголовки, которые были возвращены.

На первой вкладке, где отображается тело запроса, есть выбор нескольких вариантов отображения. Так, Pretty интересна для получения JSON-данных – программа отформатирует их в достаточно удобном формате. Если выбрать режим Raw, информация будет представлена «как есть», без каких-либо изменений. Вкладка Preview отображает сайт в том виде, в котором он открывается в браузере.

Передача параметров формы и заголовков

В отличие от GET, запрос POST передается не в ссылке на сайт, а в теле запроса. Чтобы проверить работоспособность программы, используется обращение к адресу https://postman-echo.com/post. Во время настройки на вкладке Body нужно

включить режим form-data, затем внести схожие параметры и нажать на кнопку Send.

Если взаимодействие по API требует передачи токенов авторизации, понадобится привлечь к этому HTTP-заголовки. Такой формат работы используется, например, в движке Xenforo, написанном на PHP для развертывания форумов. Для передачи в заголовке какой-либо информации нужно зайти на вкладку Headers и добавить любое имя со значением (на выбор пользователя). После отправки информации внизу окна будет отображен ответ сервера.

Передача файла в Postman

Программа Postman позволяет отправлять файлы, а не только текстовые данные, как в приведенных выше примерах. Чтобы сделать это, достаточно перейти на вкладку Body, зайти в раздел form-data и выбрать тип параметра File (вместо Text). Затем следует нажать на кнопку Select File и выбрать отправляемый файл. После отправки данных на сервер он будет виден в секции files. Ничего сложного в процедуре нет, приведенная выше схема работает со всеми типами файлов.

7. СПИСОК ВОПРОСОВ

1. Что такое GET метод?
2. Что такое POST метод?
3. Для чего нужна программа Postman?
4. Основное меню программы Postman.
5. Ключевые отличия методов POST и GET.

8. БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Дронов В. HTML и CSS. БХВ-Петербург, 2020
2. Методы HTTP –запросов [Электронный ресурс] https://ru.gadget-info.com/difference-between-get
3. Postman [Электронный ресурс] https://stfalcon.com/ru/blog/post-amp/pstman

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра информационной безопасности

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«_31_» _03_____ 2023 г.

РАЗРАБОТКА РАЗШИРЕННОГО API-СЕРВЕРА

Методические указания по выполнению практических работ для студентов укрупненной группы специальностей и направлений подготовки 10.00.00

Курск 2023

УДК 004.424

Составители: А.В. Митрофанов, А.А. Чеснокова

Рецензент

Кандидат технических наук, доцент кафедры «Информационная безопасность» А.Л. Марухленко

Разработка расширенного api-сервера: методические указания по выполнению практической работы по дисциплине «Технологии и методы программирования» / Юго-Зап. гос. ун-т; сост.: А.В. Митрофанов, А.А. Чеснокова. Курск, 2023. 8 с., Библиогр.: с. 8.

Содержат основные теоретические и практические сведения о разработке расширенного API-сервера. Указывается порядок выполнения практической работы, правила оформления и содержание отчета.

Методические указания по выполнению практических работ по дисциплине «Технологии и методы программирования», предназначены для студентов укрупненной группы специальностей и направлений подготовки 10.00.00

Текст печатается в авторской редакции

Подписано в печать _____. Формат 60×84 1/16.
Усл.печ.л. . Уч.-изд.л. . Тираж 50 экз. Заказ 220. Бесплатно
Юго–Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

СОДЕРЖАНИЕ

1. ЦЕЛЬ РАБОТЫ

Цель практической работы – разработать расширенный API-сервер.

2. ЗАДАНИЕ

Ознакомиться с теоретическим материалом. Ознакомиться с примерами решения. Расширить функционал разработанного API-сервера в лабораторной работе «Реализация базового функционала API-сервера с применением системы контроля версий GIT».

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Получить задание.
2. Изучить теоретическую часть.
3. Подготовить отчёт о выполненной работе.

4. СОДЕРЖАНИЕ ОТЧЕТА

1. Титульный лист.
2. Краткая теория.
3. Основные этапы работы.
4. Вывод.

5. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Сервер API - это установленное приложение, которое позволяет предоставлять локальные и облачные данные с помощью настраиваемых API REST.

Сервер API позволяет создавать единый API для ваших разрозненных источников данных. Используя простой интерфейсы типа "укажи и щелкни", вы можете настроить подключений и быстрого создания REST API для ведущие СУБД, популярные форматы Lle (CSV / TSV, JSON и т. Д.), Другие локальные источники (QuickBooks, MS Excel, MS Access и др.), NoSQL исходники и источники SaaS. Поскольку API-сервер установлен прямо на вашем сервере, вы можете легко предоставить локальный и удаленный доступ к API.

Сервер API генерирует API с помощью OData. протокол по умолчанию, которые сразу используется многими популярными инструментами бизнес-аналитики, отчетности и инструменты ETL вместе с наиболее распространенными разработками фреймворки и библиотеки. Как только вы сконфигурируете подключение и добавление ресурсов к серверу API, вы можете просто отправить HTTP-запросы на чтение и записывать данные. Для основных инструментов бизнес-аналитики, отчетности и ETL которые изначально поддерживают OData, этот процесс управляется в фоновом режиме. Если вы выполняете запросы программно, вы можете ссылаться на страницу API сервера API, чтобы увидеть

список доступные конечные точки и соответствующие им HTTP методы.

Сервер API автоматически создает конечные точки для открытия списка доступных ресурсов и метаданные для ресурсов, а также Конечная точка Swagger, позволяющая использовать инструменты с поддержкой Swagger и библиотеки для беспроблемной интеграции. Для каждого ресурс добавлен для ваших сконфигурированных соединений, вы получаете отдельную конечную точку, которую можно использовать для читать и записывать данные, как это разрешено базовыми данными источник. Например, если вы сконфигурируете подключения к Salesforce, вы можете работать с информацией об учетной записи из Salesforce путем передачи разные HTTP-запросы к URL-адресу, похожему на: http://address:port/api.rsc/Salesforce_Account.

Задание на практическую работу:

Расширить функционал разработанного в лабораторной работе «Реализация базового функционала API-сервера с применением системы контроля версий GIT» API-сервера.

6. СПИСОК ВОПРОСОВ

1. Что такое API?
2. Что такое API-сервер?
3. Где используют API технологии?
4. Назовите основной функционал API.
5. Что позволяет API-сервер?

7. БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Арно Лорре. Проектирование web-api. ДМК-Пресс,2020.
2. Фольдз Д.JS. Подробное руководство, Диалектика, 2012.
3. Цепочка промисов [Электронный ресурс], https://learn.javascript.ru/promise-chaining
4. Fetch API [Электронный ресурс], https://developer.mozilla.org/ru/docs/Web/API/Fetch_API
5. API server [Электронный ресурс], https://cdata.com/

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра информационной безопасности

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«_31_» _____03_____ 2023 г.

РАЗРАБОТКА РЕГЛАМЕНТА ЗАЩИЩЕННОГО ВЗАИМОДЕЙСТВИЯ ПРОГРАММНЫХ МОДУЛЕЙ СИСТЕМЫ

Методические указания по выполнению практических работ для студентов укрупненной группы специальностей и направлений подготовки 10.00.00

Курск 2023

УДК 004.424

Составители: А.В. Митрофанов, А.А. Чеснокова

Рецензент

Кандидат технических наук, доцент кафедры «Информационная безопасность» А.Л. Марухленко

Разработка регламента защищенного взаимодействия программных модулей системы: методические указания по выполнению практической работы по дисциплине «Технологии и методы программирования» / Юго-Зап. гос. ун-т; сост.: А.В. Митрофанов, А.А. Чеснокова. Курск, 2023. 10 с., Библиогр.: с. 10.

Содержат основные теоретические и практические сведения о разработке регламента защищенного взаимодействия программных модулей системы. Указывается порядок выполнения практической работы, правила оформления и содержание отчета.

Методические указания по выполнению практических работ по дисциплине «Технологии и методы программирования», предназначены для студентов укрупненной группы специальностей и направлений подготовки 10.00.00

Текст печатается в авторской редакции

Подписано в печать _____. Формат 60×84 1/16.
Усл.печ.л. . Уч.-изд.л. . Тираж 50 экз. Заказ 221. Бесплатно
Юго–Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

СОДЕРЖАНИЕ

1. ЦЕЛЬ РАБОТЫ

Цель практической работы – разработать регламент защищенного взаимодействия программных модулей системы.

2. ЗАДАНИЕ

Ознакомиться с теоретическим материалом. Ознакомиться с примерами решения. Разработать регламент защищенного взаимодействия программных модулей системы.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Получить задание.
2. Изучить теоретическую часть.
3. Разработать регламент защищенного взаимодействия программных модулей системы на основе модуля лабораторной работы «анализ структуры программных модулей с привязкой к архитектуре».

4. СОДЕРЖАНИЕ ОТЧЕТА

1. Титульный лист.
2. Краткая теория.
3. Основные этапы работы.
4. Вывод.

5. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Программный модуль (program module) — программа или отдельная ее функциональная часть, рассматриваемая как единое целое в контекстах хранения, замены, трансляции, объединения с другими программными модулями и ее загрузки в оперативную память ЭВМ. Различают объектный и загрузочный модули. Объектный модуль (object module) — элемент объектной архитектуры (object architecture или object-oriented architecture) построения программы, в которой ее основные составные части организованы в виде отдельных блоков или модулей, допускающих их компоновку, а также индивидуальное хранение и использование. Объектным модулем называют также программный модуль, полученный в результате трансляции исходной программы. Загрузочный модуль (load module) — программный модуль в относительных адресах, полученный из объектного модуля при редактировании связей в едином, принятом в данной операционной системе формате.

Защитное программирование - основано на важной предпосылке: худшее, что может сделать модуль, это принять неправильные входные данные и затем вернуть неверный, но правдоподобный результат.

Чтобы разрешить эту проблему, в начале каждого модуля помещаются проверки входных данных на соответствие их свойств атрибутам и диапазонам изменения, на полноту и осмысленность.

Программа может проверять также вводимые данные (сообщения) пользователем с клавиатуры во время работы приложения. При этом, она должна принимать любое вводное значение (сообщение) и обработать ее соответствующим образом (в крайнем случае - выдать свое сообщение и попросить пользователя ввести новое, правильное по смыслу значение (сообщение).

При этом требуется разумный подход. Если над входными данными выполнять все мыслимые проверки, то защищающая часть программы может стать настолько сложной, что ее влияние на надежность и эффективность будет не позитивным, а негативным.

Аварийные ситуации внутри программы такие, как переполнение, деление на ноль, извлечение квадратного корня или вычисление логарифма из отрицательного числа и т.д., должны быть обработаны программой. После наступления аварийной ситуации должно быть выдано соответствующее сообщение и выполнена заданная команда. Выполнять же вышеперечисленные операции лучше после проверки аргументов заданным условиям. Это займет немного времени на дополнительное программирование, но значительно повысит надежность программного обеспечения в целом.

6. ПРИМЕР ВЫПОЛНЕНИЯ РАБОТЫ

Правила связи программных модулей по информации

1. Информация зон глобальных переменных доступна дня использования любым модулем, входящим и комплекс программ или группу программ в соответствии с областью действия зоны глобальных переменных, т.е. глобальные переменные, могут быть доступны не для всего комплекса программ, а лишь для указанной в описании группы модулей.

2. Локальные переменные доступны лишь в пределах того модуля, в котором они определены или объявлены.

3. Для взаимодействия вызываемых и вызывающих модулей создаются зоны обменных переменных, информация из которых доступна лишь модулям, непосредственно связанным по управлению.

4. После окончания работы вызываемого модуля считается, что регистры не содержат информации, являющийся результатом его работы. Запрещается их использовать в вызывающем модуле.

5. Информация, находящаяся в регистрах вызывающего модуля, при вызове должна быть сохранена на период выполнения вызываемого модуля и восстановлена при возврате управления в вызывающий модуль. Сохранение регистров может осуществлять как вызывающий, так и вызываемый модуль.

Правила связи программных модулей по управлению

1. Передача управления вызываемому модулю всегда осуществляется через его начало, т.е. через первый оператор или команду.

2. Если необходимо исполнить модуль с некоторой внутренней точки, то вызов осуществляется стандартным образом (через первый оператор), а точка начала задается в виде параметра, при этом в начале вызываемого модуля должен стоять переключатель, который обеспечивает передачу управления к внутренним точкам входов по параметру, указанному при обращении.

3. Выход из вызываемого модуля всегда происходит через его естественное окончание, т.е. после нормального его завершения.

4. По окончании исполнения вызываемого модуля управление передастся в вызывающий модуль на оператор, следующий непосредственно за оператором вызова.

5. Модули низших уровней или одного уровня иерархии могут

вызываться для исполнения только моделями высших уровней, т.е. модули низших уровней не могут вызывать модули высших уровней, а модули одного уровня - вызывать друг друга.

6. В каждом модуле должна быть предусмотрена возможность подключения контрольных и отладочных средств; операторы, реализующие эти средства, обычно сосредоточиваются в конце модуля.

7. СПИСОК ВОПРОСОВ

1. Что такое программный модуль?
2. Что такое защищенное программирование?
3. Что такое объектный модуль?
4. Что такое загрузочный модуль?
5. Правила связи программных модулей по управлению.
6. Правила связи программных модулей по информации.

8. БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Дж.Хьюз, Дж.Мичтом. Структурный подход к программированию. М.: Мир, 1980. - С. 29-71.

2. В.Турский. Методология программирования. - М.: Мир, 1981. - С. 90-164.

3. Е.А.Жоголев. Технологические основы модульного программирования//Программирование,1980, #2. - С. 44-49.

4. R.C.Holt. Structure of Computer Programs: A Survey//Proceedings of the IEEE, 1975, 63(6). - P. 879-893.

5. Г.Майерс. Надежность программного обеспечения. М.: Мир, 1980. - С. 92-113.

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра информационной безопасности

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«_31_» _____03_____ 2023 г.

ФОРМАЛИЗАЦИЯ ПРЕДМЕТНОЙ ОБЛАСТИ В ВИДЕ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ

Методические указания по выполнению практических работ для студентов укрупненной группы специальностей и направлений подготовки 10.00.00

Курск 2023

УДК 004.424

Составители: А.В. Митрофанов, А.А. Чеснокова

Рецензент

Кандидат технических наук, доцент кафедры «Информационная безопасность» А.Л. Марухленко

Формализация предметной области в виде реляционной базы данных: методические указания по выполнению практической работы по дисциплине «Технологии и методы программирования» / Юго-Зап. гос. ун-т; сост.: А.В. Митрофанов, А.А. Чеснокова. Курск, 2023. 13 с., Библиогр.: с. 13.

Содержат основные теоретические и практические сведения формализации предметной области в виде реляционной базы данных. Указывается порядок выполнения практической работы, правила оформления и содержание отчета.

Методические указания по выполнению практических работ по дисциплине «Технологии и методы программирования», предназначены для студентов укрупненной группы специальностей и направлений подготовки 10.00.00

Текст печатается в авторской редакции

Подписано в печать _____. Формат 60×84 1/16.
Усл.печ.л. . Уч.-изд.л. . Тираж 50 экз. Заказ 222. Бесплатно
Юго–Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

СОДЕРЖАНИЕ

1. ЦЕЛЬ РАБОТЫ

Цель практической работы – формализировать предметную область в виде реляционной базы данных.

2. ЗАДАНИЕ

Ознакомиться с теоретическим материалом. Ознакомиться с примерами решения. Выбрать предметную область и провести ее формализацию.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Получить задание.
2. Изучить теоретическую часть.
3. Выбрать предметную область.
4. Провести формализацию области

4. СОДЕРЖАНИЕ ОТЧЕТА

1. Титульный лист.
2. Краткая теория.
3. Скриншоты БД.
4. Вывод.

5. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Понятие реляционный (англ. Relation – отношение) связано с разработками известного английского специалиста в области систем баз данных Эдгара Кодда (Edgar Codd). Эти модели характеризуются простотой структуры данных, удобным для пользователя табличным представлением и возможностью использования формального аппарата алгебры отношений и реляционного исчисления для обработки данных. Реляционная модель ориентирована на организацию данных в виде двумерных таблиц. Каждая реляционная таблица представляет собой двумерный массив и обладает следующими свойствами: каждый элемент таблицы – один элемент данных; все ячейки в столбце таблицы однородные, то есть все элементы в столбце имеют одинаковый тип (числовой, символьный и т. д.); каждый столбец имеет уникальное имя; одинаковые строки в таблице отсутствуют; порядок следования строк и столбцов может быть произвольным. Основными понятиями реляционных баз данных являются тип данных, домен, атрибут, кортеж, первичный ключ и отношение.

Тип данных в реляционной модели данных полностью адекватно понятию типа данных в языках программирования. Обычно в современных реляционных БД допускается хранение символьных, числовых данных, битовых строк, специализированных числовых данных (таких как "деньги"), а также специальных "темпоральных" данных (дата, время,

временной интервал). Понятие домена более специфично для баз данных, хотя и имеет некоторые аналогии с подтипами в некоторых языках программирования. В самом общем виде домен определяется заданием некоторого базового типа данных, к которому относятся элементы домена, и произвольного логического выражения, применяемого к элементу типа данных. Если вычисление этого логического выражения дает результат "истина", то элемент данных является элементом домена. Наиболее правильной интуитивной трактовкой понятия домена является понимание домена как допустимого потенциального множества значений данного типа. Например, домен "Имена" в схеме, показанной выше, определен на базовом типе строк символов, но в число его значений могут входить только те строки, которые могут изображать имя (в частности, такие строки не могут начинаться с мягкого знака). Кортеж, соответствующий данной схеме отношения, - это множество пар {имя атрибута, значение}, которое содержит одно вхождение каждого имени атрибута, принадлежащего схеме отношения. "Значение" является допустимым значением домена данного атрибута (или типа данных, если понятие домена не поддерживается). Тем самым, степень или "арность" кортежа, т.е. число элементов в нем, совпадает с "арностью" соответствующей схемы отношения. Попросту говоря, кортеж - это набор именованных значений заданного типа. Отношение – это множество кортежей, соответствующих одной схеме отношения. Иногда, чтобы не путаться, говорят "отношение-схема" и "отношение-экземпляр", иногда схему отношения

называют заголовком отношения, а отношение как набор кортежей - телом отношения. На самом деле, понятие схемы отношения ближе всего к понятию структурного типа данных в языках программирования. Было бы вполне логично разрешать отдельно определять схему отношения, а затем одно или несколько отношений с данной схемой. В свою очередь отношения имеют два важных свойства: арность – число атрибутов в отношении; мощность – это кардинальное число отношения, т.е. число кортежей (строк) в отношении. Каждое реляционное отношение соответствует одной сущности (объекту предметной области) и в него вносятся все атрибуты сущности. Для каждого отношения необходимо определить первичный ключ и внешние ключи. Ключ или потенциальный ключ – это минимальный набор атрибутов, по значениям которых можно однозначно выбрать требуемый экземпляр сущности. Минимальность означает, что исключение из набора любого атрибута не позволяет идентифицировать сущность по оставшимся. Каждая сущность должна, но не обязана обладать хотя бы одним возможным ключом. Другими словами ключ – это поле или набор полей, однозначно идентифицирующий запись. Значение первичного ключа в таблице БД должно быть уникальным, то есть в таблице не должно существовать двух ил более записей с одинаковым значением первичного ключа. Первичные ключи облегчают установление связей между таблицами. Поскольку первичный ключ должен быть уникальным, для него могут использоваться не все поля таблицы.

6. ПРИМЕР ВЫПОЛНЕНИЯ РАБОТЫ

Пример проектирования реляционной базы данных

В качестве примера возьмем базу данных компании, которая занимается издательской деятельностью.

Анализ предметной области

База данных создаётся для информационного обслуживания редакторов, менеджеров и других сотрудников компании. БД должна содержать данные о сотрудниках компании, книгах, авторах, финансовом состоянии компании и предоставлять возможность получать разнообразные отчёты.

В соответствии с предметной областью система строится с учётом следующих особенностей:

каждая книга издаётся в рамках контракта;

книга может быть написана несколькими авторами;

контракт подписывается одним менеджером и всеми авторами книги;

каждый автор может написать несколько книг (по разным контрактам);

порядок, в котором авторы указаны на обложке, влияет на размер гонорара;

если сотрудник является редактором, то он может работать одновременно над несколькими книгами;

у каждой книги может быть несколько редакторов, один из них

– ответственный редактор;

каждый заказ оформляется на одного заказчика;

в заказе на покупку может быть перечислено несколько книг.

Для инфологического проектирования воспользуемся методом «сущность-связь». Для того, чтобы представить, как устроена предметная область нужно задать множество объектов реального мира (главная проблема что считать объектом). Объект – семантическое понятие, которое может быть полезно при обсуждении устройств реального мира. Сущность реального мира – объекты – не обязательно материальны – важно понятие существенно и различимо для других. Между объектами могут возникать связи трех видов:

один к одному 1:1 (пациент: место в палате);

один к многим 1:n и многие к одному n:1;

многие ко многим n:n (пациент : хирург).

Выделим базовые сущности этой предметной области:

Сотрудники компании. Атрибуты сотрудников – ФИО, табельный номер, пол, дата рождения, паспортные данные, ИНН, должность, оклад, домашний адрес и телефоны. Для редакторов

необходимо хранить сведения о редактируемых книгах; для менеджеров – сведения о подписанных контрактах.

Авторы. Атрибуты авторов – ФИО, ИНН (индивидуальный номер налогоплательщика), паспортные данные, домашний адрес, телефоны. Для авторов необходимо хранить сведения о написанных книгах.

Книги. Атрибуты книги – авторы, название, тираж, дата выхода, цена одного экземпляра, общие затраты на издание, авторский гонорар.

Контракты будем рассматривать вместе с книгами, т.к. каждая книга связана непосредственно с одним контрактом. Таким образом Бд примет следующий вид(рис.1).

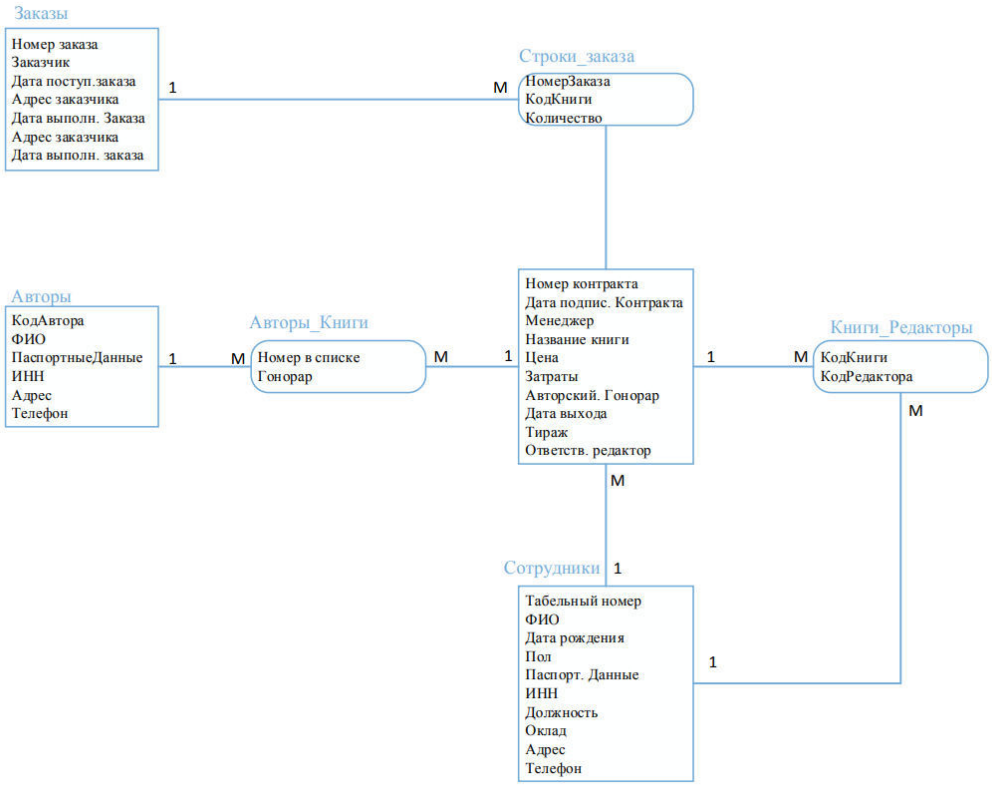

Рисунок 1 – итоговая база данных

7. ВАРИАНТЫ ЗАДАНИЙ

Музей	Пиццерия	Документооборот
Минимаркет	Прокат	Поликлиника
Строительная компания	Транспортная компания	Спортивный клуб
Картинная галерея	Книжный склад	Товары почтой
Авиакомпания	Деканат	кулинария

8. СПИСОК ВОПРОСОВ

1. Что такое предметная область?
2. Что такое база данных?
3. Что такое реляционный?
4. Что такое кортеж?
5. Что такое тип данных?

9. БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Карпов И. Базы данных. Учебное пособие, Питер, 2013. - С. 240.

2. Полищук Ю., Боровский А. Базы данных и их безопасность. Учебное пособие, ИНФА-М, 2020. - С. 210.

3. Трещев И. Базы данных. Учебное пособие, 2019. - С. 190.

4. База данных как модель предметной области [Электронный ресурс] https://skobelevserg.jimdofree.com/база-данных-как-модель-педметной-области/

5. Этап проектирования данных [Электронный ресурс] http://www.mstu.edu.ru/study/materials/zelenkov/ch_5_1.html