МИНОБРНАУКИ РОССИИ

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

Кафедра информационной безопасности

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

АНАЛИЗ СТРУКТУРЫ ПРОГРАММНЫХ МОДУЛЕЙ С ПРИВЯЗКОЙ К АРХИТЕКТУРЕ

Методические указания по выполнению лабораторных 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

Анализ структуры программных модулей с привязкой к архитектуре: методические указания по выполнению лабораторной 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 об анализе структуры программных модулей с привязкой к архитектуре. Указывается порядок выполнения лабораторной работы, правила оформления и содержание отчета.

Методические указания по выполнению лабораторных 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 211. Бесплатно
Юго–Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

СОДЕРЖАНИЕ

1. ЦЕЛЬ РАБОТЫ

Цель лабораторной работы – запустить React-приложение.

2. ЗАДАНИЕ

Ознакомиться с теоретическим материалом. Ознакомиться с примерами решения. Выбрать программный модуль и провести анализ его структуры.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Получить задание.
2. Изучить теоретическую часть.
3. Выбрать программный модуль.
4. Провести анализ выбранного модуля

4. СОДЕРЖАНИЕ ОТЧЕТА

1. Титульный лист.
2. Краткая теория.
3. Структура модуля.
4. Вывод.

5. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

5.1 Фреймворк Node.js

Язык *JavaScript* создавался в середине девяностых компанией Mozilla как простой скриптовый язык программирования, встраиваемый в код HTML-страниц. Как вспоминает его автор Брэндан Эйх, разработчики ставили перед собой цель обеспечить «язык для склеивания» составляющих частей веб-ресурса: изображений, плагинов и Java-апплетов, который был бы удобен для веб-дизайнеров и программистов, не обладающих высокой квалификацией. Он создавался для программирования как на стороне клиента, так и на стороне сервера.

Но, как часто бывает, со временем разработчики начали использовать *JavaScript* совсем иначе, чем было задумано. Самый неправильно понятый в мире язык программирования в итоге стал самым популярным. К настоящему времени он лидирует в качестве средства для разработки веб-приложений на стороне клиента. Это произошло из-за того, что эволюция веба двинулась в сторону интерактивных веб-приложений, хотя Сеть не была изначально предназначена для этого, как не был предназначен и *JavaScript.* Пришлось приспособиться.

Данная программная платформа был изначально написана Райаном Далом в 2009 году, спустя примерно 30 лет после появления первой серверной JavaScript-среды, LiveWire Pro Web от Netscape. Первоначальный выпуск поддерживал только Linux и Mac OS X. Даль раскритиковал ограниченные возможности самого популярного в 2009 году веб-сервера Apache HTTP Server для Для новой роли понадобился новый инструментарий. Постепенно *JavaScript* оброс целой инфраструктурой фреймворков, библиотек, компиляторов и протоколов. В том числе появилось несколько платформ исполнения серверных и клиентских приложений. *Node.js* — самая популярная из них. Эту платформу выпустил американский программист Райан Дал в 2009 году.

Программная платформа Node.js работает на движке V8, который транслирует JavaScript в машинный код. Грубо говоря, сам Node является приложением C++, которое получает на входе JavaScript-код и выполняет его.

В *Node* есть собственный интерфейс на *C++* для взаимодействия с устройствами ввода-вывода на компьютере. То есть эта платформа фактически превращает *JavaScript* из специализированного скриптового языка в язык общего назначения. Это означает, что на *Node.js* вы можете писать любые компьютерные программы.

Node.js представляет собой платформу для написания JavaScript-приложений с использованием внешних библиотек.

Благодаря *Node.js* написанный для браузера код *JavaScript* получает доступ к глобальным объектам, таким как document и window, наряду с другими *API* и библиотеками. С помощью *Node* код обращается к жесткому диску, базам данных и Сети. Это делает возможным написание абсолютно любых приложений: от утилит командной строки и видеоигр до полноценных веб-серверов.

Чаще всего *Node.js* используется при написании веб-приложений с интенсивным вводом-выводом. Самый распространенный пример — это веб-серверы. *Node.js* популярен для создания приложений реального времени: чатов, коммуникационных программ и игр. Многие приложения Node.js имеют и серверную, и клиентскую части.

Касательно функционала, Node.js позволяет создавать веб-серверы и сетевые инструменты, используя JavaScript, в том числе и набор «модулей», которые выполняют различные основные функции. Модули предназначены для ввода / вывода файловой системы, работы в сети (DNS, HTTP, TCP, TLS / SSL или UDP), двоичных данных (буферы), функций криптографии, потоки данных и другие основные функции. Модули Node.js используют API, разработанный для снижения уровня сложности написания серверных приложений.

JavaScript - единственный язык, который Node.js поддерживает изначально, но доступно много языков компиляции в JS. В результате приложения Node.js

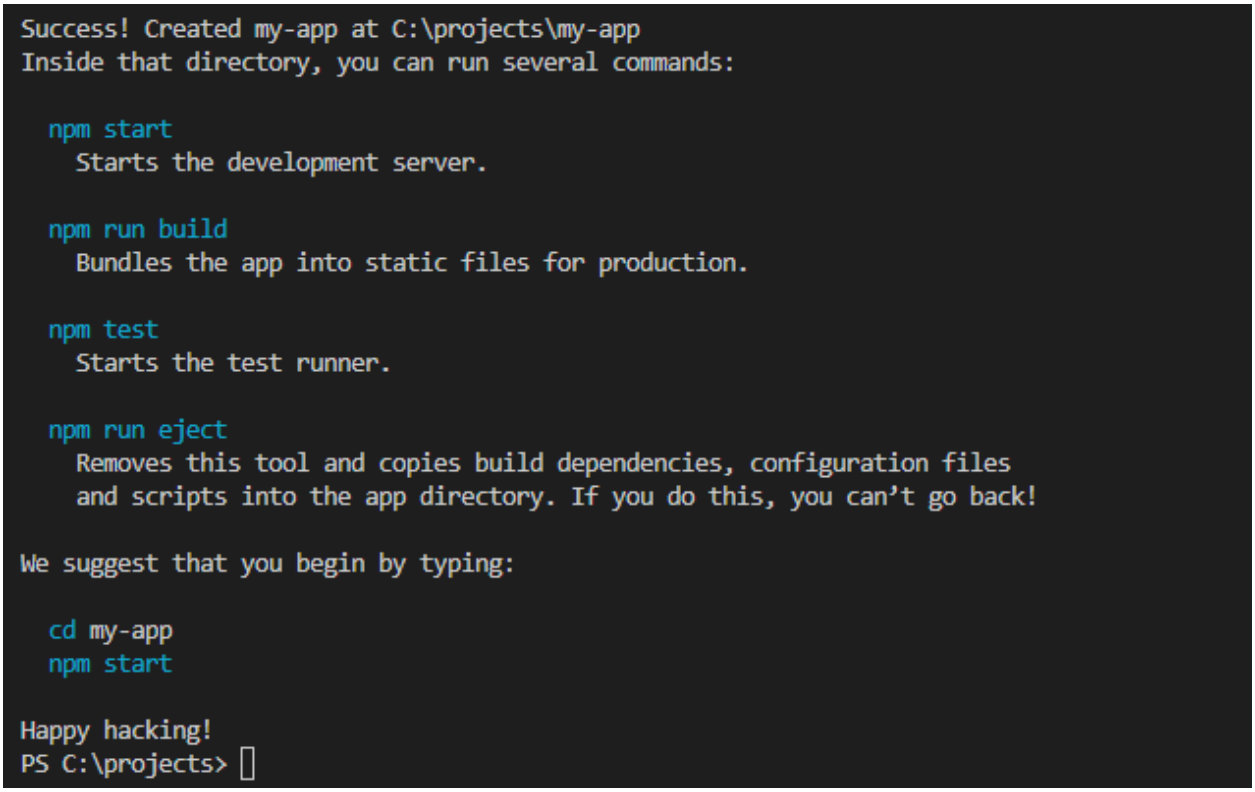
могут быть написаны на CoffeeScript, Dart , TypeScript , ClojureScript и других.
Node.js в основном используется для создания сетевых программ, такихкак веб-серверы. Наиболее существенным отличием между Node.js и PHP является то, что большинство функций в PHP блокируются до завершения (команды выполняются только после завершения предыдущих команд), тогда как функции Node.js не блокируются (команды выполняются одновременно или даже параллельно, и используют обратные вызовы для оповещения о завершении или сбое).
Node.js официально поддерживается в Linux, macOS и Microsoft

5.2 Основы фреймворка React.js

React — это библиотека JavaScript с открытым кодом для создания внешних пользовательских интерфейсов. В отличие от других библиотек JavaScript, предоставляющих полноценную платформу приложений, React ориентируется исключительно на создание представлений приложений через инкапсулированные единицы (называются компонентами), которые сохраняют состояние и генерируют элементы пользовательского интерфейса. Вы можете разместить отдельный компонент на веб-странице или вложить иерархии компонентов для создания сложного пользовательского интерфейса.
Компоненты React часто пишутся на JavaScript и JSX (JavaScript XML), который является расширением JavaScript, очень похожим на HTML и включающим некоторые функции синтаксиса для оптимизированного выполнения распространенных задач, например регистрации обработчиков событий для элементов пользовательского интерфейса. Компонент React реализует метод отрисовки, который возвращает код JSX, представляющий пользовательский интерфейс компонента. В веб-приложении код JSX, возвращаемый компонентом, преобразуется в поддерживаемый браузером код HTML, который затем обрабатывается для отображения браузером.

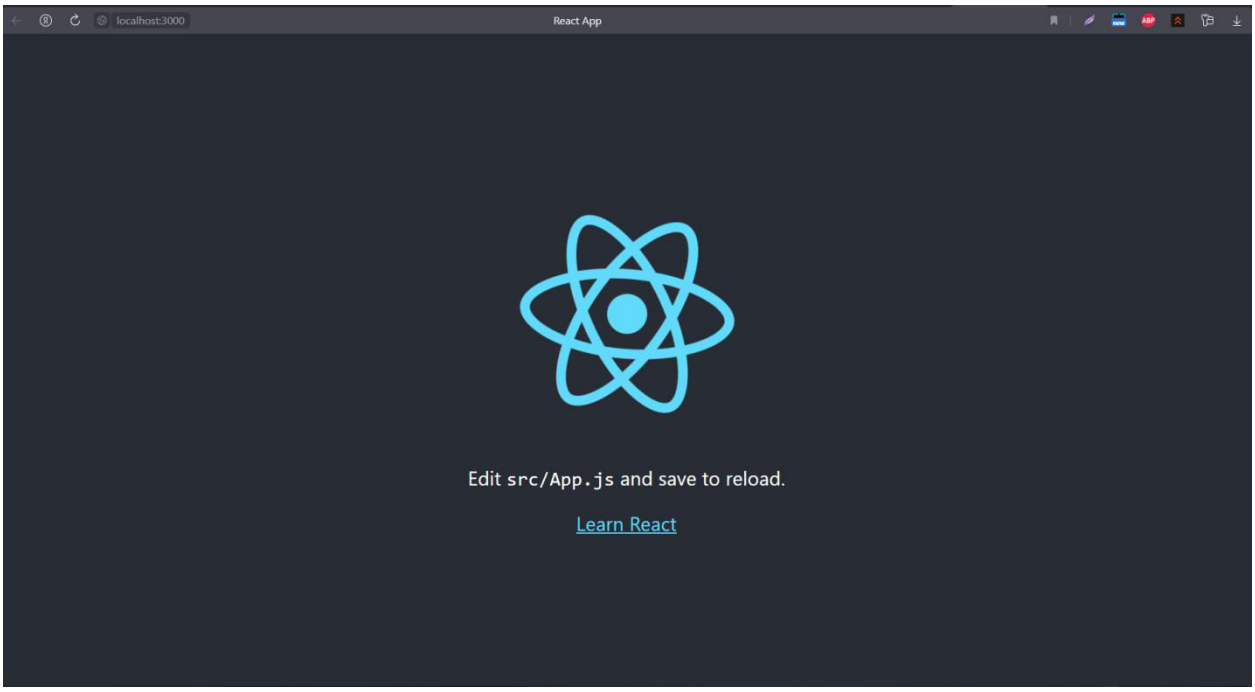
6. ПРИМЕР ВЫПОЛНЕНИЯ РАБОТЫ

1. Загружаем и устанавливаем Visual Studio Code с официального сайта.
2. Загружаем и устанавливаем фреймворк Node.js с официального сайта (при установке обязательно добавляем Node.js в путь PATH).
3. Открываем VS Code, вызываем терминал , выбираем директорию создания приложения (пример cd C:/Desktop)
4. Вписываем следующую команду: npx create-react-app «имя приложения»
5. Если все написано правильно, начнется создание приложения, загрузка стандартных библиотек , в итоге в терминале вы увидите следующее сообщение:



6. Далее запускаем приложение командой npm start

7. После чего в браузере откроется стандартное React-приложение



7. ВАРИАНТ ЗАДАНИЙ

Запустить React-приложение через Visual Studio Code .

8. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое React?
2. Что такое Node.js?
3. Какой язык программирования поддерживает Node.js изначально?
4. Какие дочерние языки программирования появились в ходе развития фреймворка Node.js?
5. Для чего используется фреймворк Node.js?
6. Перечислите основные преимущества React перед другими библиотеками JS.
7. Для чего используют модули API?
8. Какое расширение JS используется чаще всего в React?

9. 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//Программиро вание,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.

6. Я.Пайл. АДА - язык встроенных систем. М.: Финансы и статистика, 1984. - С. 67-75.

МИНОБРНАУКИ РОССИИ

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

Кафедра информационной безопасности

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

ИССЛЕДОВАНИЕ ЗАЩИЩЕННОСТИ И БЫСТРОДЕЙСТВИЯ РАБОТЫ API-ФУНКЦИЙ НА ЛОКАЛЬНОМ СЕРВЕРЕ

Методические указания по выполнению лабораторных 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

Исследование защищенности и быстродействия работы API-функций на локальном сервере: методические указания по выполнению лабораторной 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 о исследование защищенности и быстродействия работы API-функций на локальном сервере. Указывается порядок выполнения лабораторной работы, правила оформления и содержание отчета.

Методические указания по выполнению лабораторных 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 212. Бесплатно
Юго–Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

СОДЕРЖАНИЕ

1. ЦЕЛЬ РАБОТЫ

Цель лабораторной работы – научиться использовать программу Postman для тестирования API.

2. ЗАДАНИЕ

Ознакомиться с теоретическим материалом. Ознакомиться с примерами решения. Протестировать API.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Получить задание.
2. Изучить теоретическую часть.
3. Выбрать API.
4. Протестировать API

4. СОДЕРЖАНИЕ ОТЧЕТА

1. Титульный лист.
2. Краткая теория.
3. Тесты API.
4. Вывод.

5. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Что такое API?

API-интерфейсы обычно используются для передачи информации между двумя системами и функционируют либо на уровне кода, либо на уровне сети. Это зависит от того, работают ли обе системы на одном компьютере. Следует отметить важную деталь: современные API принимают форму репрезентативной передачи состояния (REST), Это обеспечивает простая функция разработки конкретного вызова API для вызова всей информации, необходимой для изменения или доступа к веб-сервису. Еще одно интересное преимущество заключается в том, что обеспечивает четкое разделение между интерфейсами (детали презентации) и Админцетр (механизм доступа к данным).

Принцины успешного тестирования безопасности

1. Критерии входов

API должен отклонять любой ввод неправильных типов (например, ноль или пустой) или неправильные размеры. У последнего есть некоторые трудности с определением диапазона допустимых входных данных, поскольку REST API должен предсказывать подход клиента.

Каждый вход имеет ожидаемый результат - и ничего другого не следует принимать. Для определенного диапазона ввода и вывода это простая задача, в отличие от ситуации, когда пользователь отправляет свой собственный контент (например, платформы чата).

Эти входные данные за пределами ожидаемых доменов не должны приниматься.

2. Предоставлять данные по мере необходимости

Как только разрешения определены, пользователям должен быть предоставлен доступ к данным только в соответствии с их требованиями. Несмотря на то, что после реализации разрешений и связанного с ними доступа к ресурсам это кажется простым, существует слишком много переменных, которые могут измениться.

Это пара параметров, на которые следует тестировать каждый API, чтобы избежать утечки данных и других компрометирующих ситуаций. Правильно спроектированный и протестированный API должен иметь возможность установить начальный уровень защиты, который должен быть одобрен протоколом тестирования безопасности.

Различные типы тестирования безопасности API

1. Процесс аудита безопасности - Идеальный процесс аудита безопасности учитывает следующие факторы в качестве основных барьеров безопасности:

- Освободи Себя условия и разрешения, необходимые пользователям для доступа к определенным уровням данных
- Уровни аутентификации требуемый пользователем для использования API и способа идентификации
- Правила шифрования для хранимых данных и экземпляров расшифровки

2. Тестирование на проникновение - Этот этап включает моделирование атак для проверки API на наличие уязвимостей, ошибочного кодирования или других угроз безопасности. Это помогает в усилении внешней безопасности и устранении ошибок разработки на начальных этапах. Благодаря сочетанию автоматизированных

(Metasploit, Acunetix и т. Д.) И ручных инструментов тестирование API можно проводить в контролируемой среде.

- Продолжить с список известных уязвимостей в API
- Установить рейтинг в соответствии с риском, который представляет каждая проблема (веб-сайт OWASP дает хорошее представление о связанных факторах риска)
- Отправляйте запросы и проектируйте атаки Помня об этих рисках безопасности как внутри, так и за пределами сети
- Держи глаза открытыми для другие вопросы которые могут появиться в процессе тестирования
- Запишите все уязвимости, обсудите меры по исправлению положения, внедрите их и проведите повторное тестирование.

3. Fuzz-тестирование - После разработки базового контрольного списка безопасности и попыток взлома пора довести API до крайности и оцените результаты. Это можно сделать с помощью большого количества одновременно отправляемых запросов (DDoS-атаки), быстрого изменения типа получаемых данных и т. Д. Это продемонстрирует другой набор уязвимостей, когда система находится под давлением, и не менее важно устранить .

Основные шаги для выполнения тестирования безопасности API

Тестирование безопасности API обычно включает отправку запросов через клиентское программное обеспечение (например, Insomnia) в конечную точку приложения, которая затем оценивается. Не забудьте использовать автоматизированное программное обеспечение для части тестирования, так как оно ускоряет задачу (можно провести аудит безопасности или нечеткое тестирование).

1. Каковы требования к безопасности вашей системы?

- Потребность в Сертификат TLS / SSL и доступный по HTTPS
- Количество уровней разрешений для доступа к разным ресурсам
- Поток аутентификации и требование к внешнему провайдеру
- Аспекты API, которые могут быть атакованы и слабые места

2. Настройка среды тестирования и проверка, все ли в порядке

В зависимости от маленькие или большие приложения, должна быть разработана промежуточная среда для тестирования. Более крупные лучше работают в отдельной среде с макетом ресурсов. Проверьте, правильно ли работает API, с помощью нескольких тестовых запросов.

3. Укажите входной домен

Прежде чем переходить к индивидуализированному тестированию, вам необходимо: определить, что означает каждый параметр и разрешены его различные комбинации. Это гарантирует, что вы правильно определите те значения, которые не разрешены, и те, которые уязвимы для атак, таких как SQL-инъекция или грубая сила.

5. Разработать и выполнить

Осталось только провести тесты и проверьте, соответствует ли полученный результат ожидаемому. Ответьте на пару вопросов, например:

- Если есть возможность загрузки файла, что делать, если появился вредоносный файл?
- Что произойдет, если в пространстве, запрашивающем информацию у пользователя, будет введен ввод HTML / JS?

- Доступ к ресурсам за пределами вашего уровня разрешений (и если к ним можно получить доступ через HTTP и HTTPS).

6. ПРИМЕР ВЫПОЛНЕНИЯ РАБОТЫ

1. Установить программу для тестирования API – Postman.
2. Выбрать API
3. Произвести тестирование выбранного API с помощью Postman.

7 ВАРИАНТЫ ЗАДАНИЙ

1. PokeAPI.
2. NASA API.
3. Open Food Facts.
4. TransLoc OpenAPI.
5. Urban Dictionary API.
6. Merriam-Webster Dictionary API.
7. Numbers API.
8. WeatherBit API.
9. US Government Data API.
10. Bible API.

8 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое API?
2. Перечислите принципы успешного тестирования безопасности.
3. Какие типы тестирования API есть?
4. Какие требования к безопасности системы существуют?

9. БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Арно Лорре. Проектирование web-api. ДМК-Пресс,2020.

2. Ричардсон Л. RESTful web APIs: [Текст] O’reilly media, 2013– 406с.

3. Керриск М. Э. LINUX API: [Текст] Питре, 2010 – 1248с.

4. Стратегия тестирования [Электронный ресурс]: https://habr.com/ru/post/568360/

5. Тестирование API [Электронный ресурс] https://gb.ru/posts/kak-testirovat-api-ili-postman-dlya-chajnikov

МИНОБРНАУКИ РОССИИ

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

Кафедра информационной безопасности

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

НАСТРОЙКА ИНТЕГРИРОВАННОЙ СРЕДЫ РАЗРАБОТКИ И СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

Методические указания по выполнению лабораторных 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

Настройка интегрированной среды разработки и системы управления базами данных: методические указания по выполнению лабораторной работы по дисциплине «Технологии и методы программирования» / Юго-Зап. Гос. ун-т; сост.: А.В. Митрофанов, А.А. Чеснокова. – Курск, 2023. – 16 с. Библиогр.: с. 16.

Содержат основные теоретические и практические сведения о настройке интегрированной среды разработки и системы управления базами данных. Указывается порядок выполнения лабораторной работы, правила оформления и содержание отчета.

Методические указания по выполнению лабораторных 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 213. Бесплатно
Юго–Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

СОДЕРЖАНИЕ

1. ЦЕЛЬ РАБОТЫ

Цель лабораторной работы – настроить среду разработки Visual Studio Code и СУБД PostgressSQL.

2. ЗАДАНИЕ

Ознакомиться с теоретическим материалом. Ознакомиться с примерами решения. Настроить среду разработки Visual Studio и СУБД SQL Server.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Получить задание.
2. Изучить теоретическую часть.
3. Настроить среду разработки Visual Studio и СУБД SQL Server

4. СОДЕРЖАНИЕ ОТЧЕТА

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

5. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Интегрированная среда разработки (IDE) — это многофункциональная программа, которая поддерживает многие аспекты разработки программного обеспечения. Интегрированная среда разработки Visual Studio — это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Помимо стандартного редактора и отладчика, которые есть в большинстве сред IDE, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для улучшения процесса разработки.

Система управления базами данных (СУБД) – это комплекс программно-языковых средств, позволяющих создать базы данных и управлять данными. Иными словами, СУБД — это набор программ, позволяющий организовывать, контролировать и администрировать базы данных. Большинство сайтов не могут функционировать без базы данных, поэтому СУБД используется практически повсеместно.

Основные функции СУБД:

- управление данными во внешней памяти (на дисках);
- управление данными в оперативной памяти с использованием дискового кэша;
- журнализация изменений (сохранение истории), резервное копирование и восстановление базы данных после сбоев;
- поддержка языков БД (язык определения данных, язык манипулирования данными).

Каждая СУБД основывается на какой-либо модели данных, это является одним из признаков классификации. По модели данных СУБД бывают:

Иерархические. В этой модели данных используется представление БД в виде древовидной структуры, состоящей из данных разных уровней.

Сетевые. Данная модель является расширением иерархического подхода. Иерархическая модель подразумевает, что запись-потомок может иметь строго одного предка, в то время как в сетевой структуре потомок может иметь любое количество предков.

Реляционные. СУБД, ориентированные на организацию данных как набор связанных записей и атрибутов в двумерной таблице.

Объектно-ориентированные. Для управления БД, основанными на объектной модели данных. Как правило основываются на объектно-ориентированных языках программирования.

Объектно-реляционные. Объединяет в себе концепции реляционной модели с дополнительными объектно-ориентированными возможностями.

SQL и реляционные БД: почему в них важно разбираться

Сегодня по-прежнему наиболее популярными при создании веб-приложений и сервисов остаются реляционные базы данных. Для управления реляционными базами данных используется язык SQL (Structured Query Language — структурированный язык запросов). Изначально SQL был инструментом работы пользователя с базой данных, однако со временем язык усложнился и стал скорее инструментом разработчика, чем конечного пользователя.

Наиболее популярные СУБД

Различные рейтинги самых популярных СУБД возглавляют Oracle, MySQL, Microsoft SQL Server, PostgreSQL.

MySQL

Считается одной из самых распространенных СУБД. MySQL — реляционная СУБД с открытым исходным кодом, главными плюсами которой являются ее скорость и гибкость, которая обеспечена поддержкой большого количества различных типов таблиц.

Кроме того, это надежная бесплатная система с простым интерфейсом и возможностью синхронизации с другими базами данных. В совокупности эти факторы позволяют использовать MySQL как крупным корпорациям, так и небольшим компаниям.

Microsoft SQL Server

Как следует из названия, фирменная СУБД, разработанная Microsoft. Оптимальная для использования в операционных системах семейства Windows, однако может работать и с Linux.

Система позволяет синхронизироваться с другими программными продуктами компании Microsoft, а также обеспечивает надежную защиту данных и простой интерфейс, однако отличается высокой стоимостью лицензии и повышенным потреблением ресурсов.

В целом, однако, сохраняет свою популярность, в немалой степени из-за того, что продукты корпорации Microsoft используются многими компаниями.

PostgreSQL

СУБД PostgreSQL — еще одна популярная и бесплатная система. Наибольшее применение нашла для управления БД веб-сайтов и

различных сервисов. Она универсальна, то есть подойдет для работы с большинством популярных платформ.

При этом PostgreSQL — объектно-реляционная СУБД, что дает ей некоторые преимущества над другими бесплатными СУБД, в большинстве являющимися реляционными.

Oracle

Первая версия этой объектно-реляционной СУБД появилась в конце 70-х, и с тех пор зарекомендовала себя как надежная, функциональная и практичная. СУБД Oracle постоянно развивается и дорабатывается, упрощая установку и первоначальную настройку и расширяя функционал.

Однако существенным минусом данной СУБД является высокая стоимость лицензии, поэтому она используется в основном крупными компаниями и корпорациями, работающими с огромными объемами данных.

6. ПРИМЕР ВЫПОЛНЕНИЯ РАБОТЫ

Чтобы начать работу, скачайте и установите Visual Studio Code. После установки требуется создать приложение, для этого требуется: Создать папку с любым именем -> Запустите Visual Studio Code.-> Создать файл index.js ->в терминале инициалищируем приложение командой npm init –y -> далее устанавливаем зависимости и прописываем следующие команду в терминале : npm install express pg pg-hstore sequelize cors dotenv, а также npm i -D nodemon . После этого появится окно примерно следующего содержания (рис 1.)

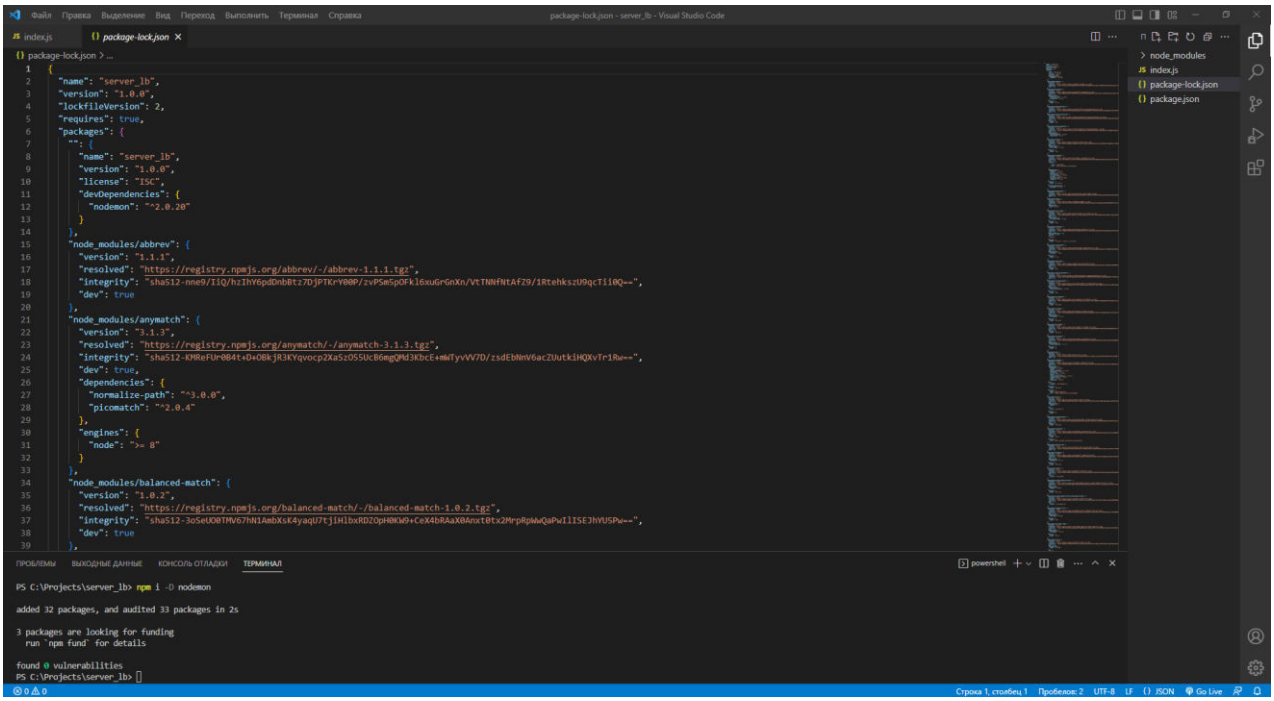


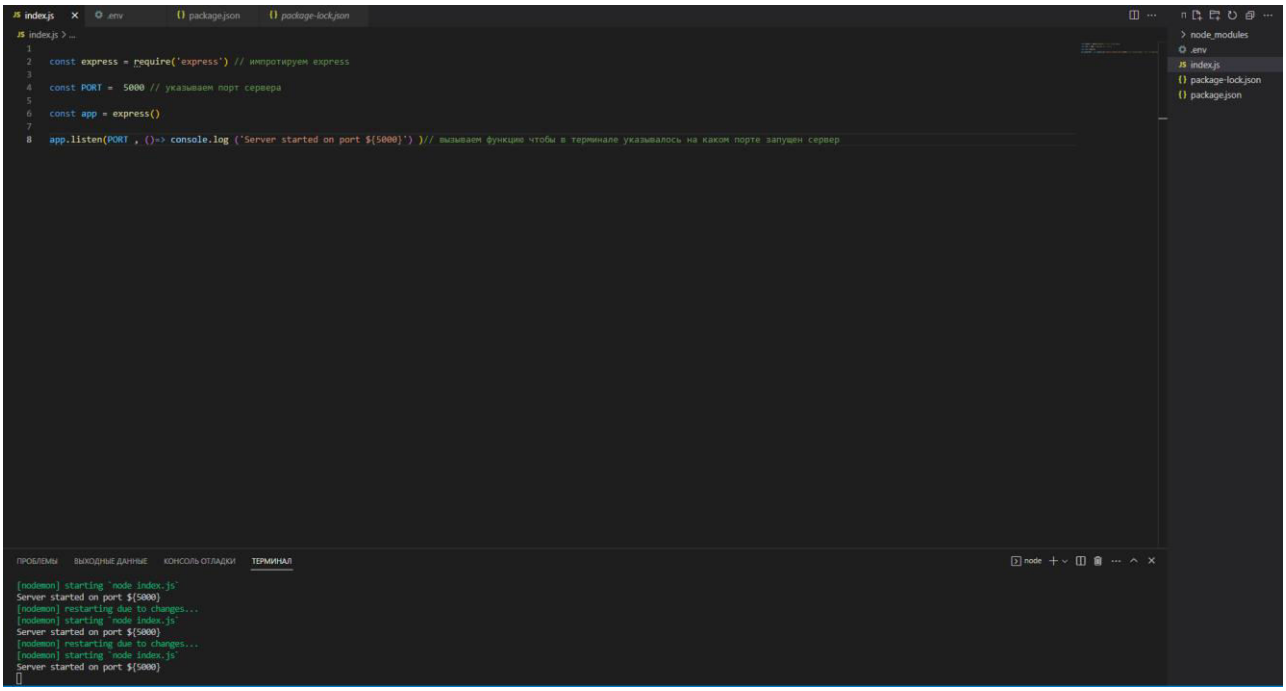
Рисунок 1 –окно Visual Studio Code

Далее в файле package.json пишем исполняемую команду в поле scripts

```
"scripts": {
    "dev" : "nodemon index.js"
  },
```

Переходим к структуре приложения

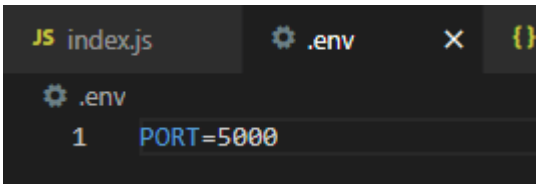
В index.js прописываем следующее



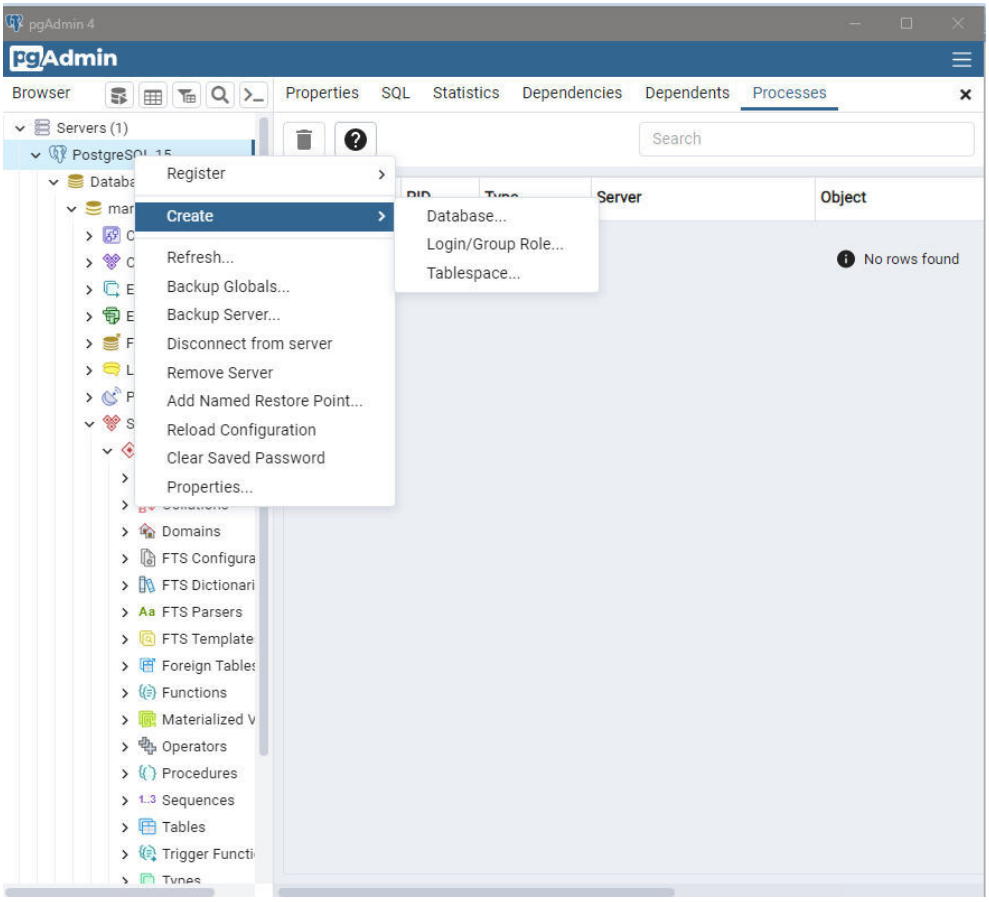
После чего запускаем наш сервер командой npm run dev

Для удобства создадим файл .env и запишем туда всю конфигурацию окружения, перепишем код index.js, занесем в env порт на котором будем запускать сервер.

```
require('dotenv').config()
const express = require('express') // импротируем express

const PORT = process.env.PORT || 5000 // указываем порт сервера

const app = express()

app.listen(PORT, () => console.log(`Server started on port ${PORT}`))// вызываем функцию чтобы в терминале указывалось на каком порте запущен сервер
```



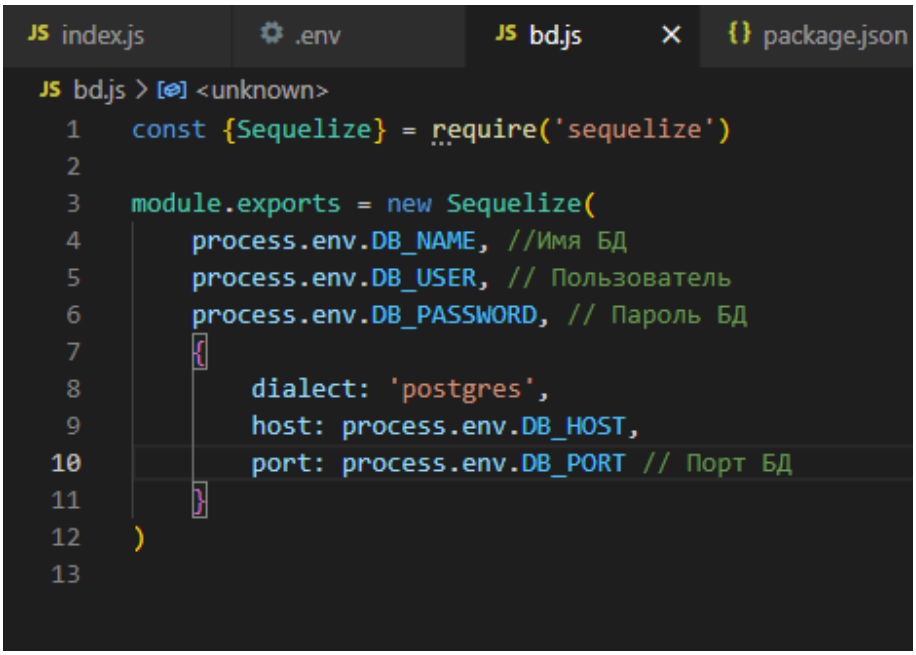
Далее устанавливаем PostgressSql, запускаем pgAdmin4, создаем новую базу данных



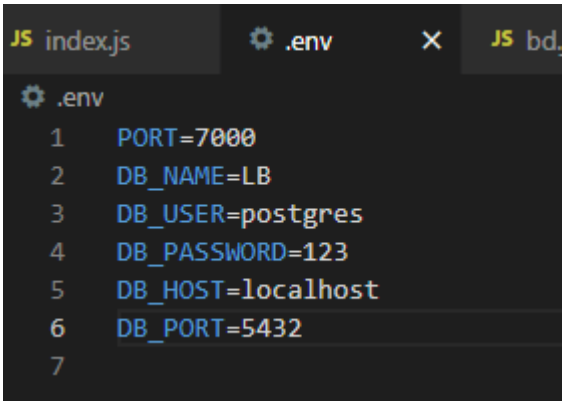
Настройка аутентификации сервера

Создаем файл BD.js , куда занесем основные настройки подключения к базе данных.

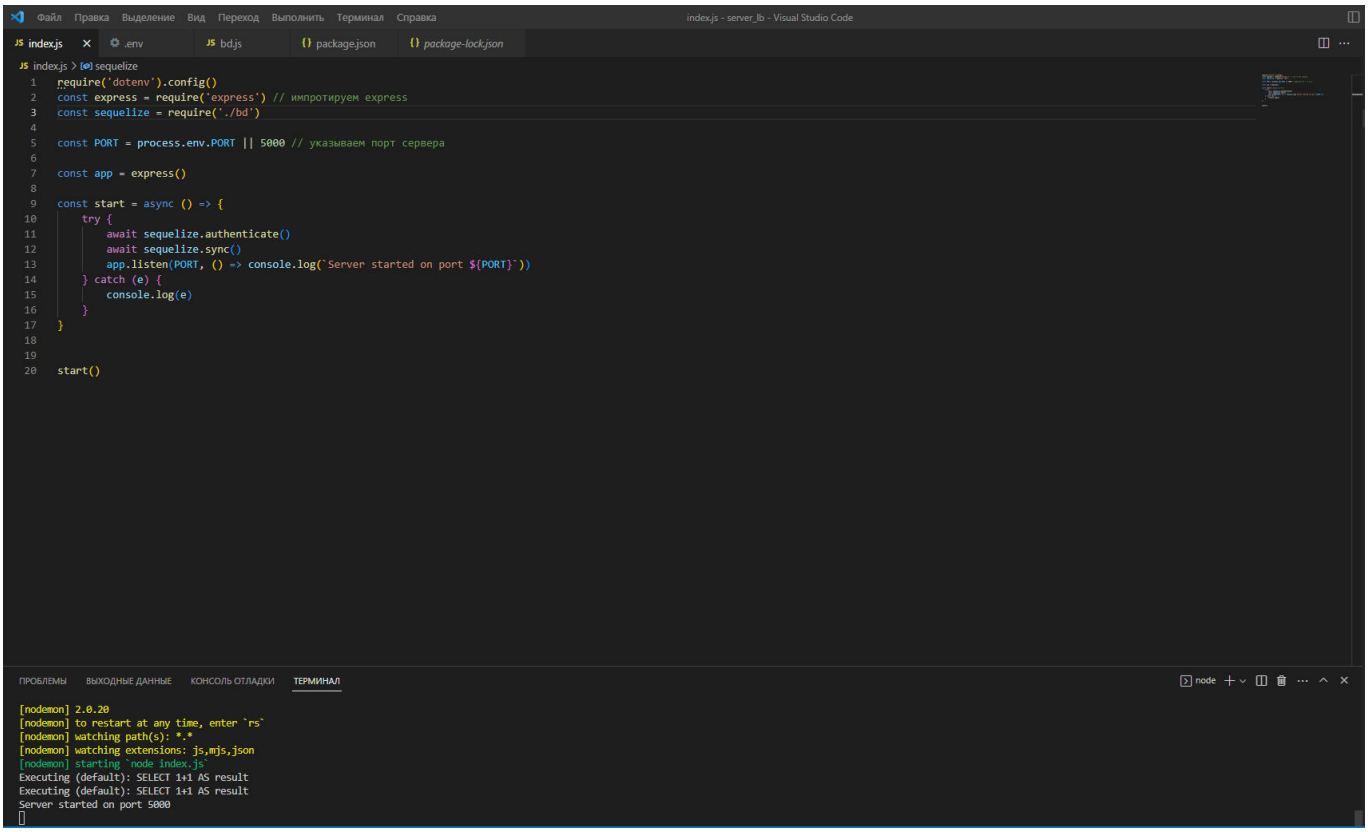
Этот файл будет иметь следующее содержание:



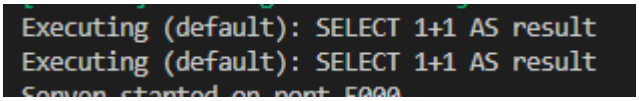
Все эти переменные запишем в файл .env



После этого импортируем наши изменения в файл index.js и получим результат



Эти значения в логах оповещают нас о том что к БД мы подключились корректно



7 ВАРИАНТ ЗАДАНИЙ

При выполнении работы, в качестве названия БД использовать свои инициалы.

8. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое интегрированная среда разработки?
2. Что такое СУБД?
3. Назовите основные функции СУЬД.
4. Какие бывают СУБД?
5. Назовите наиболее популярные СУБД.

9. БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Рендольф Н., Гарднер Д.. Visual Studio для профессионалов». М.:-Вильямс,2010.

2. Настройки интегрированной среды разработки [Электронный ресурс] https://docs.microsoft.com/ru-ru/visualstudio/ide/personalizing-the-visual-studio-ide?view=vs-2019.

3. Установка интегрированной среды разработки [Электронный ресурс] https://docs.microsoft.com/ru-ru/visualstudio/install/install-visual-studio?view=vs-2019

4. Руководство Sql Server [Электронный ресурс], https://docs.microsoft.com/ru-ru/sql/sql-server/tutorials-for-sql-server-2016?view=sql-server-ver15

МИНОБРНАУКИ РОССИИ

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

Кафедра информационной безопасности

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«31» 03 _____ 2023 г.

ПОДКЛЮЧЕНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА, КОНТРОЛЬ ОШИБОК И ОТЛАДКА ПРОГРАММЫ

Методические указания по выполнению лабораторных 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

Подключение пользовательского интерфейса, контроль ошибок и отладка программы: методические указания по выполнению лабораторной работы по дисциплине «Технологии и методы программирования» / Юго-Зап. Гос. ун-т; сост.: А.В. Митрофанов, А.А. Чеснокова. – Курск, 2023. – 15 с. Библиогр.: с. 15.

Содержат основные теоретические и практические сведения о подключение пользовательского интерфейса, контроль ошибок и отладка программы. Указывается порядок выполнения лабораторной работы, правила оформления и содержание отчета.

Методические указания по выполнению лабораторных 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 214. Бесплатно
Юго–Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

СОДЕРЖАНИЕ

1. ЦЕЛЬ РАБОТЫ

Цель лабораторной работы – получить представление о подключении интерфейса, контроле ошибок и отладчике программ VBA.

2. ЗАДАНИЕ

Ознакомиться с теоретическим материалом. Ознакомиться с отладчиком VBA.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Получить задание.
2. Изучить теоретическую часть.
3. Выполнить задание.

4. СОДЕРЖАНИЕ ОТЧЕТА

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

5. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Фронтенд — все, что браузер может читать, выводить на экран и / или запускать. То есть это HTML, CSS и JavaScript.

HTML (HyperText Markup Language) говорит браузеру, каково содержание страницы, например, «заголовок», «параграф», «список», «элемент списка».

CSS (Cascading Style Sheets) говорит браузеру, как отображать элементы, например, «после первого параграфа отступ в 20 пикселей» или «весь текст в элементе body должен быть темно-серым и написан шрифтом Verdana».

JavaScript говорит браузеру, как реагировать на некоторые взаимодействия, используя легкий язык программирования. Большинство сайтов на самом деле не используют много JavaScript, но если вы нажмете на что-то и содержимое страницы поменяется без белого мигания экрана, значит, где-то использовался JavaScript.

Бэкенд — все, что работает на сервере, то есть «не в браузере» или «на компьютере, подсоединенном к сети (обычно к Интернету), который отвечает на сообщения от других компьютеров».

Для бэкенда вы можете использовать любые инструменты, доступные на вашем сервере (который, по сути, является просто компьютером, настроенным для ответов на сообщения). Это означает, что вы можете использовать любой универсальный язык программирования: Ruby, PHP, Python, Java, JavaScript / Node, bash. Это также означает, что вы можете использовать системы управления

базами данных, такие как MySQL, PostgreSQL, MongoDB, Cassandra, Redis, Memcached.

Структура взаимодействия бэкенда и фронтенда

Сегодня существует несколько основных архитектур, определяющих, как будут взаимодействовать ваши бэкенд и фронтенд.

Серверные приложения

В этом случае HTTP-запросы отправляются напрямую на сервер приложения, а сервер отвечает HTML-страницей.

Между получением запроса и ответом сервер обычно ищет по запросу информацию в базе данных и встраивает ее в шаблон (ERB, Blade, EJS, Handlebars).

Когда страница загружена в браузере, HTML определяет, что будет показано, CSS — как это будет выглядеть, а JS — всякие особые взаимодействия.

Связь с использованием AJAX

Другой тип архитектуры использует для связи AJAX (Asynchronous JavaScript and XML). Это означает, что JavaScript, загруженный в браузере, отправляет HTTP-запрос (XHR, XML HTTP Request) изнутри страницы и (так сложилось исторически) получает XML-ответ. Сейчас для ответов также можно использовать формат JSON.

Это значит, что у вашего сервера должна быть конечная точка, которая отвечает на запросы JSON- или XML-кодом. Два примера протоколов, используемых для этого — REST и SOAP.

Клиентские (одностраничные) приложения

AJAX позволяет вам загружать данные без обновления страницы. Больше всего это используется в таких фреймворках, как Angular и Ember. После сборки такие приложения отправляются в браузер, и любой последующий рендеринг выполняется на стороне клиента (в браузере).

Такой фронтенд общается с бэкендом через HTTP, используя JSON- или XML-ответы.

Универсальные/изоморфные приложения

Некоторые библиотеки и фреймворки, например, React и Ember, позволяют вам исполнять приложения как на сервере, так и в клиенте.

В этом случае для связи фронтенда с бэкендом приложение использует и AJAX, и обрабатываемый на сервере HTML.

Контроль ошибок — комплекс методов обнаружения и исправления ошибок в данных при их записи и воспроизведении или передаче по линиям связи.

Контроль целостности данных и исправление ошибок — важные задачи на многих уровнях работы с информацией (в частности, физическом, канальном, транспортном уровнях сетевой модели OSI) в связи с тем, что в процессе хранения данных и передачи информации по сетям связи неизбежно возникают ошибки. Различные области применения контроля ошибок диктуют различные требования к используемым стратегиям и кодам.

В системах связи возможны несколько стратегий борьбы с ошибками:

- обнаружение ошибок в блоках данных и автоматический запрос повторной передачи повреждённых блоков — этот подход применяется, в основном, на канальном и транспортном уровнях;
- обнаружение ошибок в блоках данных и отбрасывание повреждённых блоков — такой подход иногда применяется в системах потокового мультимедиа, где важна задержка передачи и нет времени на повторную передачу;
- упреждающая коррекция ошибок добавляет к передаваемой информации такие дополнительные данные, которые позволяют исправить ошибки без дополнительного запроса.

В контроле ошибок, как правило, используется помехоустойчивое кодирование — кодирование данных при записи или передаче и декодирование при считывании или получении при помощи корректирующих кодов, которые и позволяют обнаружить и, возможно, исправить ошибки в данных. Алгоритмы помехоустойчивого кодирования в различных приложениях могут быть реализованы как программно, так и аппаратно.

Отладка — этап разработки компьютерной программы, на котором обнаруживают, локализуют и устраняют ошибки. Чтобы понять, где возникла ошибка, приходится:

- узнавать текущие значения переменных;
- выяснять, по какому пути выполнялась программа.

Существуют две взаимодополняющие технологии отладки.

Использование отладчиков — программ, которые включают в себя пользовательский интерфейс для пошагового выполнения программы: оператор за оператором, функция за функцией, с остановками на

некоторых строках исходного кода или при достижении определённого условия.

Вывод текущего состояния программы с помощью расположенных в критических точках программы операторов вывода — на экран, принтер, громкоговоритель или в файл. Вывод отладочных сведений в файл называется журналированием.

6. ПРИМЕР ВЫПОЛНЕНИЯ РАБОТЫ

1. Открыть новый документ EXCEL или WORD.
2. Добавить в проект модуль с именем **Отладка**.
3. Добавить в модуль процедуру, подсчитывающую отношение суммы элементов массива и их произведения, а также нахождение среднего арифметического элементов массива, значение которых являются четными числами:

```
Public Sub pr_Otl1()
Dim b(10) As Integer
Dim s As Single
Dim p As Integer
Dim s1 As Single
Dim k As Integer
p = 1
For i = 1 To 10
b(i) = InputBox("Введите число")
If b(i) Mod 2 = 0 Then s1 = s1 + b(i): k = k + 1
s = s + b(i): p = p* b(i)
Next
s = s / p : MsgBox s
s1 = s1 / k : MsgBox s1
End Sub
```

4. Выполнить отладку процедуры. Запустить процедуру на выполнение и ввести значения элементов массива 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 .

VBA выдает ошибку выполнения № 6 - "переполнение" (Overflow).

В окне с сообщением об ошибке нажать кнопку **Debug (Отладка)**. Цветом выделяется оператор, выполнение которого привело к возникновению ошибки.

5. Просмотреть значения переменных, влияющих на результат в режиме

прерывания. Для этого в VBA имеется несколько средств:

- для просмотра значения конкретной переменной, установить на ее имя указатель мыши и значение переменной отобразится во всплывающей подсказке. Если этого не произошло, следует установить флажок **Auto Data Tips** на вкладке **Editop** в окне команды **Tools/Options.** Просмотреть значение переменных p и b(i);
- для просмотра значений всех локальных переменных выполнить команду **View\Locals Window (Вид\Окно локальных переменных);**
- для просмотра значений нужных переменных и выражений использовать окно **Watches (Контрольные значения)**.

Просмотреть в контрольном окне значение переменных p и b(i), а также выражения p*b(i). Для добавления переменной (выражения) в контрольное окно, следует выделить ее имя (выражение) в операторе и выполнить команду **Add Watch (Отладка\Добавить контрольное значение).**

Просмотр окна **Watches** позволяет сделать вывод, что тип переменной p не соответствует помещаемому значению. Тип переменной p необходимо изменить на Long (Длинное целое). Внести изменение в текст процедуры. Закрыть окно **Watches.**

Повторно выполнить программу.

6. Продолжить отладку программы. Запустить программу на выполнение и ввести значения 1, 2, 3, 4, 5, 6, 7, 8, 9, 0. VBA выдает ошибку выполнения №11 - "деление на 0" (Division by zero). Просмотреть значение переменных, участвующих в вычислениях в операторе, выполнение которого привело к возникновению ошибки.

7. Перейти из режима прерывания программы в режим конструктора. Для этого выполнить команду **Run\Reset (Запуск\Сброс)** или нажать кнопку **Сброс** на панели **Стандарт.**

8. Продолжить отладку программы. Запустить ее на выполнение и ввести значения 1,3,5,7,9,11,13,15,17, 19. VBA выдает ошибку выполнения № 6 - "переполнение". Просмотреть значение переменных, участвующих в вычислениях

в операторе, выполнение которого привело к возникновению ошибки.

9. Добавить в процедуру обработчик ошибок. Для этого отредактировать процедуру в соответствии с приведенным ниже текстом:

```
Public Sub pr_Otl1()
Dim b(10) As Integer
Dim s As Single
Dim p As Long
Dim s1 As Single
Dim k As Integer
On Error GoTo MyErrHandler
p = 1
Vvod: For i = 1 To 10
b(i) = InputBox("Введите число")
If b(i) Mod 2 = 0 Then s1 = s1 + b(i): k = k + 1
s = s + b(i): p = p * b(i)
Next
s = s / p
MsgBox s
s1 = s1 / k
MsgBox s1
Exit Sub
MyErrHandler:
Select Case Err.Number
Case 6
MsgBox "среди элементов массива нет четных чисел, повторите ввод"
Err.Clear: s1 = 0: p = 1
Resume Vvod
Case 11
MsgBox "Один из элементов массива равен нулю," & _
" отношение суммы к произведению элементов найти нельзя"
```

```
Resume Next
Case Else
Dim err_num As Integer
err_num = Err.Number
MsgBox "Ошибка выполнения" & err_num
Err.Clear
Resume Next
End Select
End Sub
```

10. Выполнить отладку обработчика ошибок. Для этого выполнить программу несколько раз, вводя по очереди такие исходные данные, которые привели бы к возникновению всех обрабатываемых ошибок.

11. Осуществить пошаговое выполнение программы (трассировку) с помощью команды **Debug\Step Into (Отладка \ Шаг с заходом),** анализируя изменение значений переменных процедуры на каждом шаге выполнения**.**

12. Осуществить пошаговое выполнение части программы. Для этого необходимо:

- установить точку останова, щелкнув указателем мыши по левому полю рядом с нужной строкой программы, или выполнив команду **Debug\Toggle Breakpoint (Отладка\Точка останова),** или нажав клавишу F9 (например: установить точку останова на строке s = s / p);
- запустить процедуру на выполнение;
- после прерывания работы процедуры просмотреть значение переменных s,s1,p,k;
- продолжить выполнение процедуры в режиме трассировки;
- удалить точку останова

7 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое тест и как выполняется тестирование?
2. Перечислите виды ошибок, возникающих в процессе создания и эксплуатации программного обеспечения.
3. Какие ошибки могут возникнуть на этапе выполнения программы? В чем причина возникновения этих ошибок?
4. Какие существуют способы контроля над значениями переменных?
5. Как можно изменить значения переменных в процессе отладки программы?

8. БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Бычков М.И. Основы программирования на VBA для Microsoft Excel:[Текст] учебное пособие, 2010 - 99с.

2. Кукушкина Е.В. Начальные сведения о языке программирования Visual Basic for Application: [Текст] учебная литература для ВУЗов, 2014 – 111с.

3. Абрамян М. Э. Практикум по параллельному программированию с использованием электронного задачника Programming Taskbook for MPI: [Текст] учебное пособие, 2010 – 172с.

4. Паппас, К. Х. Отладка в C++ [Текст] : руководство для разработчиков / Пер. с англ.; Под ред. В. В. Тимофеева. - М. : БИНОМ, 2001. - 512 с.

5. Этапы создания программы на языке Ассемблер. Изучение интегрированной среды отладчика Turbo Debug [Электронный ресурс] : методические указания к лабораторной работе №1 / Юго-Зап. гос. ун-т ; сост.: Е. А. Титенко, Д. И. Родионов, Е. А. Петрик. - Курск : ЮЗГУ, 2010. - 14 с.

МИНОБРНАУКИ РОССИИ

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

Кафедра информационной безопасности

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

РАЗРАБОТКА CRUD ПРИЛОЖЕНИЕ НА БАЗЕ WEB-ФРЕЙМВОРКА

Методические указания по выполнению лабораторных 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 CRUD приложения на базе web-фреймворка: методические указания по выполнению лабораторной 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 о разработке CRUD приложения на базе web-фреймворка. Указывается порядок выполнения лабораторной работы, правила оформления и содержание отчета.

Методические указания по выполнению лабораторных 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 215. Бесплатно
Юго–Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

СОДЕРЖАНИЕ

1. ЦЕЛЬ РАБОТЫ

Цель лабораторной работы – разработать CRUD-приложение на базе web-Фреймворка.

2. ЗАДАНИЕ

Ознакомиться с теоретическим материалом. Ознакомиться с примерами решения. Разработать CRUD-приложение на базе web-Фреймворка.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Получить задание.
2. Изучить теоретическую часть.
3. Разработать CRUD-приложение.

4. СОДЕРЖАНИЕ ОТЧЕТА

1. Титульный лист.
2. Краткая теория.
3. Описать основные этапы разработки.
4. Вывод.

5. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Фреймворк — набор инструментов для ускоренной разработки сайта, нацелен на решение определенных задач.

Говоря простыми словами, framework – это каркас, который состоит из множества различных библиотек, которые облегчают разработку программного продукта или сайта. То есть фреймворк это набор из нескольких библиотек. Фреймворки различны по своим возможностям и функциям. Их наполнение зависит от поставленных задач. Они нужны для того, чтобы программист или дизайнер могли сосредоточиться на уникальных задачах, а не заново изобретали колесо.

Веб-фреймворк — это каркас для написания веб-приложений. Он определяет структуру, задаёт правила и предоставляет необходимый набор инструментов для разработки.

Типы веб-фреймворков

Классифицировать фреймворки для веб-приложений можно по двум основаниям: задачам, которые они решают, и размеру.

Бэкенд-фреймворки

Это фреймворки веб-разработки, которые работают на серверной стороне. В основном они отвечают за отдельные, но критически важные части приложения, без которых оно не сможет нормально работать. Вот несколько самых популярных фреймворков, а также языки, с которыми они работают:

- Django — Python;
- Symfony, Laravel — PHP;
- Express.js — JavaScript;

- Ruby on Rails — Ruby.

Правила и архитектура серверных фреймворков не даёт возможности разработать веб-приложение с богатым интерфейсом. Они ограничены в своей функциональности, однако вы всё равно можете создавать простые страницы и разные формы. Также они могут формировать выходные данные и отвечать за безопасность в случае атак.

Фронтенд-фреймворки

Фронтенд-фреймворки отвечают за внешний вид веб-приложения. В отличие от серверных, они никак не связаны с логикой работы. Этот тип фреймворков работает в браузере. С их помощью можно улучшать и внедрять новые пользовательские интерфейсы, создавать разные анимации и одностраничные приложения. Вот некоторые из них:

- Angular;
- Vue.js;
- Svelte;

React — формально это не фреймворк, а библиотека, но значение этого инструмента так велико, что его постоянно сравнивают с другими веб-фреймворками.

Все эти инструменты используют JavaScript.

Фуллстек-фреймворки

Если фреймворк решает задачи и на серверной, и на клиентской стороне, то он относится к категории фуллстек. В качестве примера можно назвать Meteor. Обе его стороны — серверная и клиентская — работают на JavaScript. Поэтому вы можете создавать и использовать для них один и тот же код. Следующая особенность — «режим реального

времени». Когда вы что-то меняете в одном интерфейсе, изменения происходят и в остальных.

К фуллстек также относятся фреймворки Next.js и Nuxt. Первый создан поверх React.js, а второй работает на базе Vue.js. Такие веб-фреймворки могут быть сложными для начинающих.

Классическое приложение для работы с БД обычно называют CRUD - по первым буквам стандартных операций, Create, read, update and delete (Создание чтение обновление удаление).

Это как гамма в музыке - любое приложение состоит из этих базовых элементов. В нашем случае это веб-страница, которая выводит список записей в БД в виде списка (R), позволяет добавлять в список новые записи (C),редактировать их (U) и удалять (D).

6. ПРИМЕР ВЫПОЛНЕНИЯ РАБОТЫ

С помощью класса такое приложение реализуется буквально в несколько строчек. Также очень удобно то, что с помощью данного кода можно редактировать любые таблицы, меняя только переменные $table и $fields (ну и шаблоны, разумеется).

```
<?php
include 'safemysql.class.php';

$db     = new SafeMysql();
$table  = "users";
$fields = ['name', 'car', 'sex'];

if($_SERVER['REQUEST_METHOD']=='POST')
{
   $data = $db->filterArray($_POST, $fields);
   if (isset($_POST['delete']))
   {
      $db-
>query("DELETE FROM ?n WHERE id=?i", $table, $_POST['delete']);
   } elseif ($_POST['id']) {
      $db-
>query("UPDATE ?n SET ?u WHERE id = ?i", $table, $data, $_POST['id']);
   } else {
      $db->query("INSERT INTO ?n SET ?u", $table, $data);
   }
   header("Location: http://".$_SERVER['HTTP_HOST'].$_SERVER['PHP_S
ELF']);
   exit;
}

if (!isset($_GET['id']))
{
   $LIST = $db->getAll("SELECT * FROM ?n", $table);
   include 'list.php';
```

```
} else {
   if ($_GET['id'])
   {
      $row = $db-
>getRow("SELECT * FROM ?n WHERE id=?i", $table, $_GET['id']);
   } else {
      $row['name'] = '';
      $row['sex']  = '';
      $row['car']  = '';
      $row['id']   = 0;
   }
   include 'form.php';
}

// эта функция не нужна для работы с БД, но нужна для примеров вывода
// поскольку вывод всегда должен быть экранирован
function e($str)
{
  return htmlspecialchars($str, ENT_QUOTES, 'utf-8');
}
```

плюс файлы шаблонов,

list.php

```
<a href="?id=0">Add item</a>
<table border=1>
   <tr>
      <td><b>Name</b></td>
      <td><b>Car</b></td>
      <td><b>Sex</b></td>
      <td><b>Action</b></td>
   </tr>
<?php foreach ($LIST as $row): ?>
   <tr>
      <td><?=e($row['name'])?></td>
      <td><?=e($row['car'])?></td>
      <td><?=e($row['sex'])?></td>
```

```
        <td><a href="?id=<?=e($row['id'])?>">Edit</a></td>
    </tr>
<?php endforeach ?>
</table>
```

и form.php

```
<form method="POST">
    Name: <input type="text" name="name" value="<?=e($row['name'])?>"><b
r>
    Car: <input type="text" name="car" value="<?=e($row['car'])?>"><br>
    Sex: <select name="sex">
        <option<?php if ($row['sex'] == 'male'):   ?> selected="selected"<?php en
dif ?>>male</option>
        <option<?php if ($row['sex'] == 'female'): ?> selected="selected"<?php en
dif ?>>female</option>
    </select>
    <input type="hidden" name="id" value="<?=e($row['id'])?>">
    <input type="submit"><br>
</form>
<a href="?">Return to the list</a>

<?php if ($row['id']): ?>
<form method="POST">
<input type="hidden" name="delete" value="<?=e($row['id'])?>">
<input type="submit" value="Delete"><br>
</form>
<? endif ?>
```

7 ВАРИАНТ ЗАДАНИЙ

В качестве варианта выбирать любой знакомый язык программирования и соответствующий ему фреймворк.

8 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое Фреймворк?
2. Что такое web-фреймворк?
3. Назовите по каким признакам можно разделить web-фреймворки.
4. Что такое бэкенд-фреймворки?
5. Что такое фронтенд-фреймворки?
6. Что такое фуллстек-фреймворки?

9. БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Дакетт Д. HTML и CSS. Эксмо,2018.
2. Макграт М. PHP-7. Эксмо-Пресс, 2019.
3. Фреймворк [Электронный ресурс], https://artjoker.ua/ru/big-brain/glossary/framework/
4. CRUD приложение [Электронный ресурс], https://qna.habr.com/q/538933.
5. Веб-фреймворки [Электронный ресурс], https://tproger.ru/translations/web-frameworks-how-to-get-started/

МИНОБРНАУКИ РОССИИ

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

Кафедра информационной безопасности

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«31» 03 2023 г.

РАЗРАБОТКА АДАПТИВНОГО ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА НА БАЗЕ WEB-ТЕХНОЛОГИЙ

Методические указания по выполнению лабораторных 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 адаптивного пользовательского интерфейса на базе web-технологий: методические указания по выполнению лабораторной работы по дисциплине «Технологии и методы программирования» / Юго-Зап. Гос. ун-т; сост.: А.В. Митрофанов, А.А. Чеснокова. – Курск, 2023. – 32 с. Библиогр.: с. 32.

Содержат основные теоретические и практические сведения о разработке адаптивного пользовательского интерфейса на базе web-технологии. Указывается порядок выполнения лабораторной работы, правила оформления и содержание отчета.

Методические указания по выполнению лабораторных 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 216. Бесплатно
Юго–Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

СОДЕРЖАНИЕ

1. ЦЕЛЬ РАБОТЫ

Цель лабораторной работы – разработать интерфейс сайта.

2. ЗАДАНИЕ

Ознакомиться с теоретическим материалом. Ознакомиться с примерами решения. Выбрать тематику задания, разработать интерфейс сайта.

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить теоретическую часть.
2. Выбрать тематику сайта.
3. Разработать адаптивный интерфейс сайта

.

4. СОДЕРЖАНИЕ ОТЧЕТА

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

5. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Адаптивный интерфейс пользователя — Adaptive user interface

Адаптивный пользовательский интерфейс (также известный как AUI) представляет собой пользовательский интерфейс (UI) , который адаптируется, то есть изменения, его расположение и элементы к потребностям пользователя или контекста , и это аналогично изменяемое каждому пользователю.

Эти взаимно обратные качества и адаптации и быть адаптированы являются, в истинном AUI, также врожденные к элементам, которые содержат компоненты интерфейса произвольных; части интерфейса могут адаптироваться и влиять на другие части интерфейса.

Это позже механизм обычно используется для интеграции двух логически различных компонентов, таких как интерактивного документа и приложения (например , в веб — браузере) в единое целое.

Адаптация пользователей часто переговоры процесс, так как дизайнеры адаптивного пользовательского интерфейса игнорируют, где компоненты пользовательского интерфейса должны идти, предоставив средства, с помощью которых как дизайнеры и пользователь могут определить их размещение, часто (хотя и не всегда) в полу- автоматизирован, если не полностью автоматизированный способ.

AUI в основном создаются на основе характеристик системы и уровнях знаний о пользователях, которые будут использовать его.

преимущества

Преимущества адаптивного пользовательского интерфейса находятся в его способности, чтобы соответствовать потребностям пользователей. Свойства с AUI позволяют показывать только необходимую информацию на основе текущего пользователя. Это создает меньше путаницы для менее опытных пользователей и обеспечивает легкость доступа по всей системе.

В зависимости от поставленной задачи, может повысить стабильность системы.

Недостатки

Трудоемкий процесс.

AUI должна быть разработана с учетом различных уровней реализации в виде, и быть в сочетании с образом, чтобы измерить любые конкретные потребности пользователей.

Требуется знание цели пользователя для того, чтобы наиболее эффективно адаптироваться. Кроме того, чтобы быть проблемой при сортировке и интерпретации информации от пользователя для того, чтобы прогнозировать их потребности, это может привести к проблемам безопасности. Поскольку информация хранится, пользователи не имеют конфиденциальность при использовании в AUI.

Типы

Адаптивный интерфейс пользователя может быть реализован различными способами. Эти реализации могут отличаться от

объема информации, доступного для определенных пользователей, или как пользователи используют приложение.

Адаптивное представление

Цель за адаптивное представление для отображения определенной информации на основе текущего пользователя. Это может означать, что пользователи только с базовыми знаниями систем будут показаны только минимальной информацией. С другой стороны, пользователь с углубленными знаниями будет иметь доступ к более подробной информации и возможностям.

Путь, что AUI может достичь этой дифференциации может быть, чтобы скрыть информацию, которая будет представлена на основе уровня опыта пользователя. Другая возможность состоит в том, чтобы контролировать количество ссылок на соответствующие источники на этой странице.

Адаптивная навигация

Адаптивная навигация намерена направлять пользователь к их конкретной цели в рамках системы, изменяя способ системы, переключение на основе определенных факторов пользователя. Эти факторы могут включать уровень пользователей знаний системы / субъект, текущую цель в рамках системы, а также другие соответствующие факторы.

Примеры адаптивной навигации могут быть достигнуты разными способами, похожих на адаптивную презентацию. Эти примеры могут включать в себя такие, как предоставление ссылок для достижения конкретной цели пользователя, давая ссылку на

страницу, где пользователь, или изменения ресурсов, доступных пользователю.

Применение в промышленности

Адаптационные пользовательские интерфейсы могут быть использованы в любой ситуации, когда пользователь будет извлечь выгоду из персонализированного интерфейса. Одним из наиболее распространенных реализаций месте в AUI в медицинской промышленности. AUI используется для дифференциации и указать, какую информацию следует показать, какой тип пользователя. Например, пациент будет показан различный уровень детализации, чем врач или медсестра.

6. ПРИМЕР ВЫПОЛНЕНИЯ РАБОТЫ

На рисунке ниже представлен пример адаптивного интерфейса сайта(рис 1).

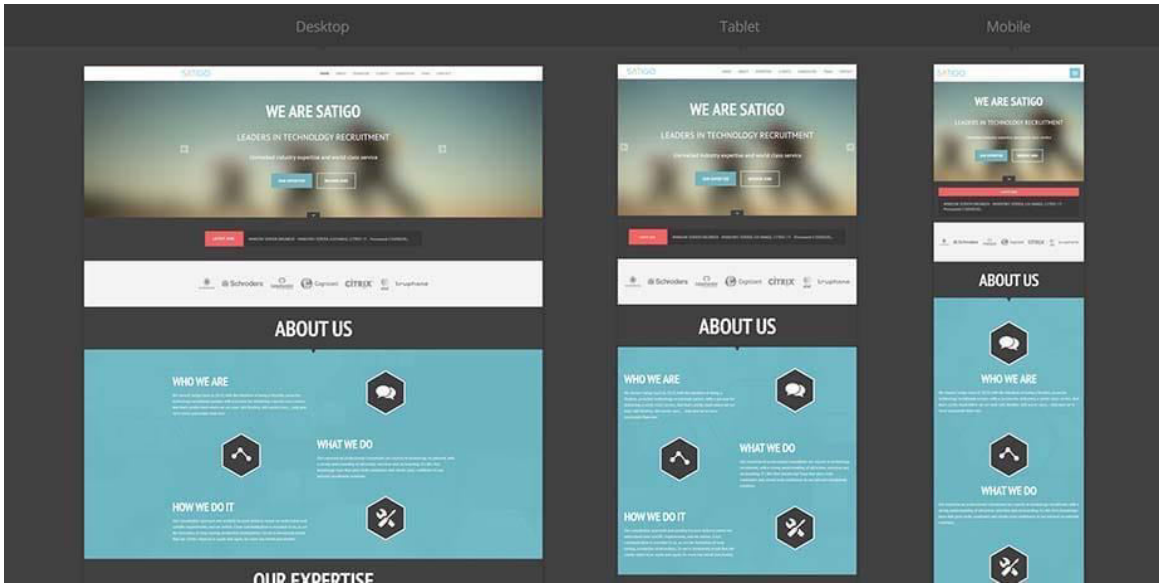


Рисунок 1 – адаптивный интерфейс

1. Метатеги и раздел <head>

Добавим в раздел <head> необходимые файлы — ссылку на используемые шрифты, библиотеку jQuery, а также плагин prefixfree (чтобы не писать для свойств браузерные префиксы):

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
```

<title>Адаптивная вёрстка сайта</title>

<link rel="stylesheet" type="text/css" href="https://fonts.googleapis.com/css?family=Open+Sans:400,400italic,600,600italic,700,700italic|Playfair+Display:400,700&subset=latin,cyrillic">

<link rel="stylesheet" type="text/css" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.4.0/css/font-awesome.css">

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

<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/2.2.2/jquery.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/prefixfree/1.0.7/prefixfree.min.js"></script>

</head>

<body>

2. Шапка страницы

В шапке страницы <header> поместим следующие элементы-контейнеры:

- логотип <a class="logo">;
- кнопку для показа/скрытия главного меню ;
- главное меню <ul id="menu">;
- форму поиска по сайту <form id="searchform">.

```
<header>
  <nav class="container">
   <a class="logo" href="">
    <span>L</span>
    <span>O</span>
    <span>G</span>
    <span>O</span>
   </a>
   <div class="nav-toggle"><span></span></div>
   <form action="" method="get" id="searchform">
    <input type="text" placeholder="Искать на сайте...">
    <button type="submit"><i class="fa fa-search"></i></button>
   </form>
   <ul id="menu">
    <li><a href="">Блог</a></li>
    <li><a href="">Портфолио</a></li>
    <li><a href="">Об авторе</a></li>
   </ul>
  </nav>
 </header>
```

3. Блок с кратким содержанием статьи

Анонс статей обернём элементом <article id="post-1" class="post">:

```
<div class="container">
 <div class="posts-list">
```

```
<article id="post-1" class="post">
 <div class="post-image"><a href=""><img src="https://html5book.ru/wp-content/uploads/2016/05/rasskaz_slovar_rodnoy_prirodi.jpg"></a></div>
 <div class="post-content">
  <div class="category"><a href="">Дизайн</a></div>
  <h2 class="post-title">Весна</h2>
  <p>Очень богат русский язык словами, относящимися к временам года и к природным явлениям, с ними связанным.</p>
  <div class="post-footer">
   <a class="more-link" href="">Продолжить чтение</a>
   <div class="post-social">
    <a href="" target="_blank"><i class="fa fa-facebook"></i></a>
    <a href="" target="_blank"><i class="fa fa-twitter"></i></a>
    <a href="" target="_blank"><i class="fa fa-pinterest"></i></a>
   </div>
  </div>
 </div>
</article>
<article id="post-2" class="post">
 ...
</article>
```

```
</div> <!-- конец div class="posts-list"-->
```

4. Боковая колонка

В боковую колонку <aside> добавим список категорий, последние записи и форму подписки на рассылку:

```
<aside>
 <div class="widget">
  <h3 class="widget-title">Категории</h3>
  <ul class="widget-category-list">
   <li><a href="">Дизайн</a> (2)</li>
   <li><a href="">Вёрстка</a> (5)</li>
   <li><a href="">Видео</a> (1)</li>
  </ul>
 </div>
 <div class="widget">
  <h3 class="widget-title">Последние записи</h3>
  <ul class="widget-posts-list">
   <li>
    <div class="post-image-small">
     <a href=""><img src="https://html5book.ru/wp-content/uploads/2016/05/rasskaz_slovar_rodnoy_prirodi.jpg"></a>
    </div>
    <h4 class="widget-post-title">Весна</h4>
   </li>
   <li>
    <div class="post-image-small">
```

```
            <a href=""><img src="https://html5book.ru/wp-content/uploads/2016/05/rasskaz_Russia.jpg"></a>
            </div>
            <h4 class="widget-post-title">Лето</h4>
           </li>
           <li>
            <div class="post-image-small">
            <a href=""><img src="https://html5book.ru/wp-content/uploads/2016/05/rasskaz_rodnaya_priroda_osen.jpg"></a>
            </div>
            <h4 class="widget-post-title">Осень</h4>
           </li>
          </ul>
         </div>
         <div class="widget">
          <h3 class="widget-title">Подписка на рассылку</h3>
          <form action="" method="post" id="subscribe">
           <input type="email" name="email" placeholder="Ваш email" required>
           <button type="submit"><i class="fa fa-paper-plane-o"></i></button>
          </form>
         </div>
        </aside>
        </div> <!-- конец div class="container"-->
```

5. Нижний колонтитул

В подвале страницы разместим информацию о копирайте, кнопки социальных сетей и ссылку на электронную почту:

```
<footer>
 <div class="container">
  <div class="footer-col"><span>Мой блог © 2016</span></div>
  <div class="footer-col">
   <div class="social-bar-wrap">
    <a title="Facebook" href="" target="_blank"><i class="fa fa-facebook"></i></a>
    <a title="Twitter" href="" target="_blank"><i class="fa fa-twitter"></i></a>
    <a title="Pinterest" href="" target="_blank"><i class="fa fa-pinterest"></i></a>
    <a title="Instagram" href="" target="_blank"><i class="fa fa-instagram"></i></a>
   </div>
  </div>
  <div class="footer-col">
   <a href="mailto:admin@yoursite.ru">Написать письмо</a>
  </div>
 </div>
</footer>
<script>
$('.nav-toggle').on('click', function(){
$('#menu').toggleClass('active');
```

```
});
</script>
</body>
</html>
```

6. Общие CSS-стили

Общие стили, сброс стилей браузера по умолчанию:

```
*, *:after, *:before {
  box-sizing: border-box;
  padding: 0;
  margin: 0;
  transition: .5s ease-in-out;
  /* добавим плавность переходов для всех элементов страницы*/
}
ul {
  list-style: none;
}
a {
  text-decoration: none;
  outline: none;
}
img {
  display: block;
  width: 100%;
}
h1, h2, h3, h4, h5, h6 {
```

```
    font-family: 'Playfair Display';
    font-weight: normal;
    letter-spacing: 1px;
  }
  body {
    font-family: 'Open Sans', arial, sans-serif;
    font-size: 14px;
    line-height: 1;
    color: #373737;
    background: #f7f7f7;
  }
  /* добавим очистку потока для всех контейнеров, внутри которых задано обтекание дочерних элементов */
  header:after, .container:after, footer:after, .widget-posts-list li:after, #subscribe:after {
    content: "";
    display: table;
    clear: both;
  }
  /* стилевой класс, который управляет шириной контейнера сетки*/
  .container {
    margin: 0 auto;
    width: 100%;
    max-width: 960px;
    padding: 0 15px;
```

```
}
7. Стили для шапки и её содержимого
header {
  width: 100%;
  background: white;
  box-shadow: 3px 3px 1px rgba(0, 0, 0, .05);
  padding: 15px 0;
  margin-bottom: 30px;
  position: relative;
}
/* логотип */
.logo {
  display: block;
  float: left;
}
.logo span {
  color: white;
  display: inline-block;
  width: 30px;
  height: 30px;
  line-height: 30px;
  border-radius: 50%;
  margin: 5px 0;
  text-align: center;
  text-shadow: 2px 2px 1px rgba(0, 0, 0, .4);
}
```

```
.logo span:nth-child(odd) {
  background: #EF5A42;
}
.logo span:nth-child(even) {
  background: #F8B763;
}
/* меню */
#menu {
  float: right;
}
#menu li {
  display: inline-block;
  margin-right: 30px;
}
#menu a {
  color: #111;
  text-transform: uppercase;
  letter-spacing: 1px;
  font-weight: 600;
  display: block;
  line-height: 40px;
}
#menu a:hover {
  color: #EF5A42;
}
#menu li:last-child {
```

```
  margin-right: 0;
}
/* форма поиска */
#searchform {
  float: right;
  margin-left: 46px;
  display: inline-block;
  position: relative;
}
#searchform input {
  width: 170px;
  float: left;
  border: none;
  padding-left: 10px;
  height: 40px;
  overflow: hidden;
  outline: none;
  color: #9E9C9C;
  font-style: italic;
}
#searchform button {
  background: transparent;
  height: 40px;
  border: none;
  position: absolute;
  right: 10px;
```

```
    color: #EF5A42;
    cursor: pointer;
    font-size: 18px;
  }
  #searchform input:focus {
    outline: 2px solid #EBEBE3;
  }
  /* кнопка переключения меню, появляющаяся при ширине 768px */
  .nav-toggle {
    display: none;
    position: relative;
    float: right;
    width: 40px;
    height: 40px;
    margin-left: 20px;
    background: #EF5A42;
    cursor: pointer;
  }
  .nav-toggle span {
    display: block;
    position: absolute;
    top: 19px;
    left: 8px;
    right: 8px;
    height: 2px;
```

```
  background: white;
}
.nav-toggle span:before, .nav-toggle span:after {
  content: "";
  position: absolute;
  display: block;
  left: 0;
  width: 100%;
  height: 2px;
  background: white;
}
.nav-toggle span:before {
  top: -10px;
}
.nav-toggle span:after {
  bottom: -10px;
}
/* класс, который будет добавлен в верхнему меню при нажатии на кнопку и покажет скрытое меню*/
#menu.active {
  max-height: 123px;
}
```

8. Стили для блока с основным содержимым

```
/* левый контейнер */
.posts-list {
  margin-bottom: 30px;
```

```
  width: 64%;
  float: left;
}
/* блок для статьи */
.post {
  margin-bottom: 35px;
}
.post-content p {
  line-height: 1.5;
  padding-bottom: 1em;
}
.post-image {
  margin-bottom: 30px;
}
.category {
  margin-bottom: 15px;
}
.category a {
  color: #F8B763;
  text-transform: uppercase;
}
.post-title {
  margin-bottom: 12px;
  font-size: 26px;
}
```

```
/* блок с кнопкой "продолжить чтение" и кнопками социальных сетей */
.post-footer {
  border-top: 1px solid #EBEBE3;
  border-bottom: 1px solid #EBEBE3;
  position: relative;
  margin-top: 15px;
}
.more-link {
  position: relative;
  display: inline-block;
  font-size: 10px;
  text-transform: uppercase;
  color: white;
  line-height: 44px;
  padding: 0 22px;
  background: #3C3D41;
  letter-spacing: 0.1em;
  white-space: nowrap;
}
.more-link:after {
  content: '';
  display: block;
  position: absolute;
  width: 0;
  height: 0;
```

```
    top: 0;
    right: 0;
    border: solid transparent;
    border-width: 22px 18px;
    border-left-color: #3C3D41;
    transform: translateX(100%);
  }
  .post-social {
    position: absolute;
    left: auto;
    top: 50%;
    right: 0;
    text-align: right;
    transform: translateY(-50%);
    padding: 0;
    font-size: 12px;
  }
  .post-social a {
    display: inline-block;
    margin-left: 8px;
    color: #F8B763;
    width: 25px;
    height: 25px;
    line-height: 23px;
    text-align: center;
    border-radius: 50%;
```

```
  border: 1px solid;
}
9. Стили для боковой колонки
/* правый контейнер */
aside {
  width: 33%;
  float: right;
}
/* блок для виджетов */
.widget {
  padding: 20px 15px;
  background: white;
  font-size: 13px;
  margin-bottom: 30px;
  box-shadow: 3px 3px 1px rgba(0, 0, 0, .05);
}
.widget-title {
  font-size: 18px;
  padding: 10px;
  margin-bottom: 20px;
  text-align: center;
  border: 2px solid #F8B763;
  box-shadow: 3px 3px 0 0 #F8B763;
}
.widget-category-list li {
  border-bottom: 1px solid #EBEBE3;
```

```
  padding: 10px 0;
  color: #c6c6c6;
  font-style: italic;
}
.widget-category-list li:last-child {
  border-bottom: none;
}
.widget-category-list li a {
  color: #626262;
  margin-right: 6px;
  font-style: normal;
}
.widget-category-list li a:before {
  content: "\f105";
  display: inline-block;
  font-family: 'FontAwesome';
  margin-right: 10px;
  color: #c6c6c6;
}
.widget-posts-list li {
  border-top: 1px solid #EBEBE3;
  padding: 15px 0;
}
.widget-posts-list li:nth-child(1) {
  border-top: none;
}
```

```
.post-image-small {
  width: 30%;
  float: left;
  margin-right: 15px;
}
.widget-post-title {
  float: left;
}
/* форма подписки */
#subscribe {
  position: relative;
  width: 100%;
  padding: 15px 0;
}
#subscribe input {
  width: 100%;
  display: block;
  float: left;
  border: 2px solid #EBEBE3;
  padding: 0 0 0 10px;
  height: 40px;
  position: relative;
  outline: none;
  color: #9E9C9C;
  font-style: italic;
}
```

```
#subscribe button {
  padding: 0 15px;
  background: transparent;
  height: 40px;
  border: none;
  position: absolute;
  right: 0;
  color: #EF5A42;
  cursor: pointer;
  font-size: 18px;
}
#subscribe input:focus+button {
  background: #EF5A42;
  color: white;
}
```

7. ВАРИАНТ ЗАДАНИЙ

№ Варианта	Задание
1	Разработать интерфейс сайта пиццерии
2	Разработать интерфейс сайта турагентства
3	Разработать интерфейс сайта путеводителя по достопримечательностям Курска
4	Разработать интерфейс сайта доставки роллов/ ресторана
5	Разработать интерфейс сайта ЮЗГУ
6	Разработать интерфейс сайта личного блога
7	Разработать интерфейс сайта со спортивными тренировками
8	Разработать интерфейс онлайн магазина
9	Разработать интерфейс сайта юридической кампании
0	Разработать интерфейс сайта любого завода

Последняя цифра в номере по списку - номер варианта.

8 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое адаптивный пользовательский интерфейс?
2. Перечислите преимущества AUI.
3. Перечислите недостатки AUI.
4. Перечислите типы AUI.

9. БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Влейминк И. Интерфейс «человек компьютер». Москва Мир,1990.

2. Монтеро М. Адаптивность и адаптивные интерфейсы на основе web. IGI Global, 2005.

3. Круг Стив, Веб-юазбилити и здравый смысл. 3-у издание. Эксмо. 2017.

4. Унгер Расс, Чендлер Кэролайн, UX-дизайн. Символ-Плюс. 2011.

5. Адаптивный интерфейс [Электронный ресурс], https://вебджем.рф/adaptiv-2/

МИНОБРНАУКИ РОССИИ

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

Кафедра информационной безопасности

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

РЕАЛИЗАЦИЯ БАЗОВОГО ФУНКЦИОНАЛА API-СЕРВЕРА С ПРИМЕНЕНИЕМ СИСТЕМЫ КОНТРОЛЯ ВЕРСИЙ GIT

Методические указания по выполнению лабораторных 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

Реализация базового функционала API-сервера с применением системы контроля версий GIT: методические указания по выполнению лабораторной 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 реализация базового функционала API-сервера с применением системы контроля версий GIT. Указывается порядок выполнения лабораторной работы, правила оформления и содержание отчета.

Методические указания по выполнению лабораторных 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 214. Бесплатно
Юго–Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

СОДЕРЖАНИЕ

1. ЦЕЛЬ РАБОТЫ

Цель лабораторной работы – научиться использовать API в простых программах

2. ЗАДАНИЕ

Ознакомиться с теоретическим материалом. Ознакомиться с примерами решения. Выбрать свой вариант задания, создать программу , использующую API-функции

3. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Получить задание.
2. Изучить теоретическую часть.
3. Выбрать значение секрета.
4. Рассчитать пороговую схему разделения секрета между n сторонами.

4. СОДЕРЖАНИЕ ОТЧЕТА

1. Титульный лист.
2. Краткая теория.
3. Расчет схемы разделения секрета.
4. Вывод.

5. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

API

API (application programming interface) — интерфейс прикладного программирования, набор методов, классов, библиотек, функций, обеспечивающих взаимодействие программ между собой.

Эта разработка была создана для упрощения труда программистов. Если попытаться дать определение простыми словами, то API — это удобный инструмент, представляющий собой набор классов, функций, процедур, стандартов, которые позволяют приложениям эффективно взаимодействовать между собой. Программисты используют этот механизм при создании самых разных систем.

Где используются API-технологии

Быстрая регистрация в приложениях через аккаунты в социальных сетях. С помощью специального API-протокола в социальных сетях Facebook, «ВКонтакте» и на других платформах пользователь получает возможность использовать упрощённый доступ к продуктам компании, пройдя быструю регистрацию на сайте через авторизацию своего аккаунта.

Google. Эта система использует интерфейс программирования для предоставления разработчикам различных приложений доступа и возможности интеграции информации на разных сервисах. Например,

можно найти и просмотреть видеоролик с платформы YouTube прямо в приложении.

Предоставление API в виде готового продукта. Разработчики предлагают доступ к своему приложению для получения оперативных данных по метеорологическим сводкам в любой точке земного шара и пр.

Основной функционал API

Популярность API обусловлена простотой применения и функциональностью. Программисту не обязательно изучать внутренние механизмы действия этого интерфейса программирования, достаточно просто применять его для объединения приложений в единую систему.

Принцип работы механизма API состоит в организации многоуровневой иерархии, в которой подчинённые компоненты создаются с одинаковой структурой. Выстраивается стандартная сетевая модель OSI с определённым количеством ступеней (не менее 7). Внутренние уровни классифицируются, выделяют приложения HTTP, IMAP, физические уровни трансляции и пр. Такое построение позволяет использовать в интерфейсе функционал нижних API для работы верхних.

Для эффективной организации работы создаются библиотеки функций и классов с описанием сигнатур и семантики. Сигнатура в данном случае является частью объявления функции, которая идентифицирует элемент. Её можно представить с помощью различных языков программирования и определить возможности перезагрузки. Описывая языки вызова, специалисты разделяют сигнатуры вызова и

реализации заданных функций. Сигнатуру определяют, учитывая область видимости и последовательностей фактических типов аргументов. Такие компоненты позволяют компилятору распознавать функции при работе с языком C++. Если это метод определённого класса, сигнатуру включают в имя данного класса.

Семантика функции описывает её действие и принципы работы. Она описывает результат вычислений и характеристики, от которых зависит его получение. То есть в таких моделях результат зависит не только от аргументов, но и от реального состояния. При этом не так и важно, что API-соединение даёт возможность получать информацию.

Библиотеки используются при написании программ и приложений, создании сервисов для обслуживания клиентов и многого другого.

API поисковых систем и веб-специалистов

Мастера, которые занимаются программированием и оформлением сайтов, а также их продвижением, используют специальный Web API. Это интерфейсы, которые включают комплект определённых HTTP-запросов. При получении такого рода запросов модуль генерирует HTTP-ответы определённой структуры. Чтобы передавать информацию, между ними применяются форматы XML или JSON. В этой сфере применения Web API является синонимом веб-службы, определёнными программами с соответствующими интерфейсами. Чтобы получить доступ к данным модулям, нужно пройти процедуру идентификации в Интернете по онлайн-адресу. То

есть при необходимости передачи данных на сервер нужно использовать серверный модуль взаимодействия с API.

Что такое Git и зачем он нужен?

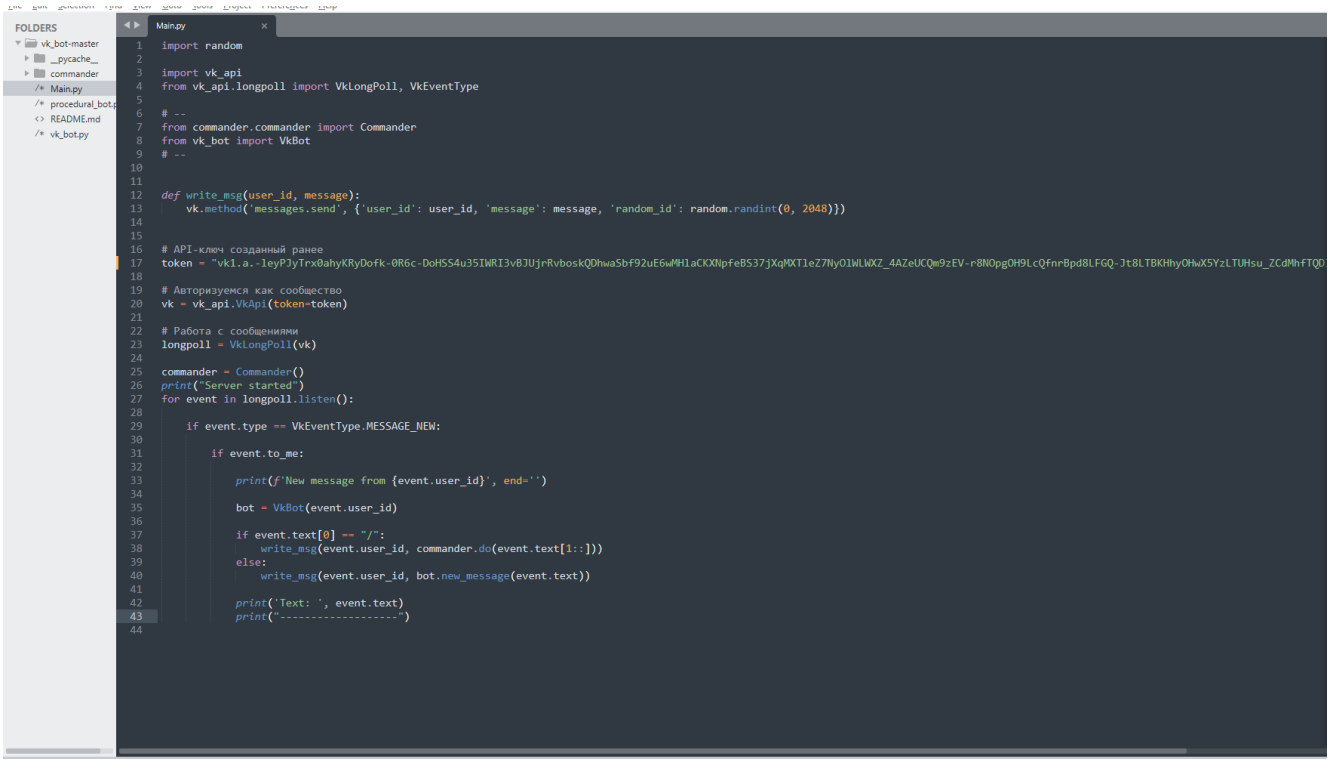
Git - это консольная утилита, для отслеживания и ведения истории изменения файлов, в вашем проекте. Чаще всего его используют для кода, но можно и для других файлов. Например, для картинок - полезно для дизайнеров.

С помощью Git-а вы можете откатить свой проект до более старой версии, сравнивать, анализировать или сливать свои изменения в репозиторий.

Репозиторием называют хранилище вашего кода и историю его изменений. Git работает локально и все ваши репозитории хранятся в определенных папках на жестком диске.

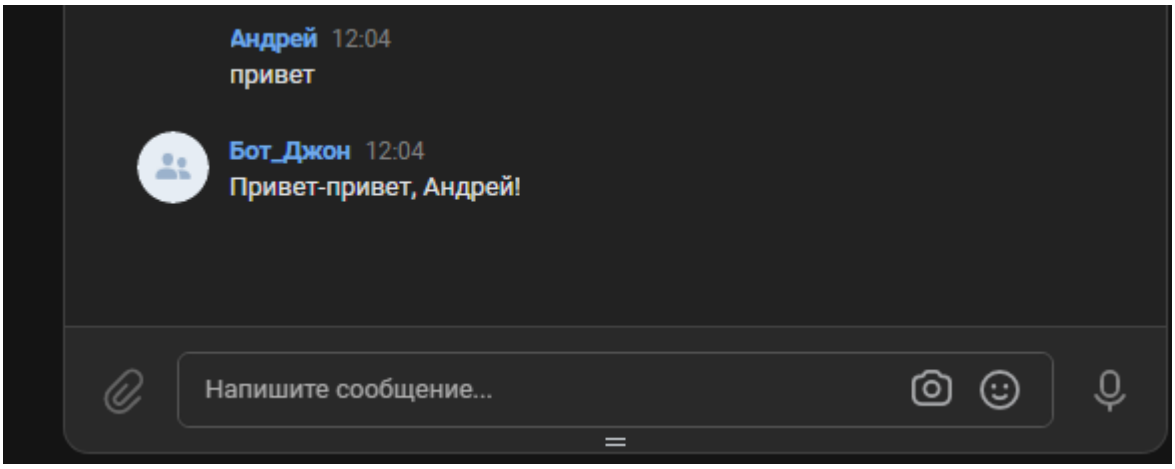
6. ПРИМЕР ВЫПОЛНЕНИЯ РАБОТЫ

Далее представлен код программы , которая создает бота ВК с помощью API на Python.



```
import random

import vk_api
from vk_api.longpoll import VkLongPoll, VkEventType

# --
from commander.commander import Commander
from vk_bot import VkBot
# --


def write_msg(user_id, message):
    vk.method('messages.send', {'user_id': user_id, 'message': message, 'random_id': random.randint(0, 2048)})


# API-ключ созданный ранее
token = "vk1.a.-leyPJyTrx0ahyKRyDofk-0R6c-DoHSS4u35IWRI3vBJUjrRvboskQDhwaSbf92uE6wMH1aCKXNpfeBS37jXqMXT1eZ7NyO1WLWXZ_4AZeUCQm9zEV-r8NOpgOH9LcQfnrBpd8LFGQ-Jt8LTBKHhyOHwX5YzLTUHsu_ZCdMhfTQD

# Авторизуемся как сообщество
vk = vk_api.VkApi(token=token)

# Работа с сообщениями
longpoll = VkLongPoll(vk)

commander = Commander()
print("Server started")
for event in longpoll.listen():

    if event.type == VkEventType.MESSAGE_NEW:

        if event.to_me:

            print(f'New message from {event.user_id}', end='')

            bot = VkBot(event.user_id)

            if event.text[0] == "/":
                write_msg(event.user_id, commander.do(event.text[1::]))
            else:
                write_msg(event.user_id, bot.new_message(event.text))

            print('Text: ', event.text)
            print("------------------")
```

Получаем результат



7 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое API?
2. Где используют API технологии?
3. Назовите основной функционал API.
4. Что такое Git?
5. Что такое репозиторий?

8. БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Арно Лорре. Проектирование web-api. ДМК-Пресс,2020.
2. Фольдз Д.JS. Подробное руководство, Диалектика, 2012.
3. Что такое GIT [Электронный ресурс], https://www.atlassian.com/ru/git/tutorials/what-is-git#version-control-with-git
4. Цепочка промисов [Электронный ресурс], https://learn.javascript.ru/promise-chaining
5. Fetch API [Электронный ресурс], https://developer.mozilla.org/ru/docs/Web/API/Fetch_API