МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ОРГАНИЗАЦИИ И ПРОВЕДЕНИЮ ПРОИЗВОДСТВЕННОЙ ПРАКТИКИ (ПРЕДДИПЛОМНОЙ)

Нижний Новгород
2026

УДК 33(07)
ББК 65.291р3
М 545

М 545 Методические рекомендации по организации и проведению производственной практики (преддипломной)/ сост. Е.Г.Неумоина. Н. Новгород: Мининский университет. 2026. 26 с.

Методические рекомендации по организации и проведению производственной практики (преддипломной) предназначены для студентов очной формы обучения по специальности 09.02.12 Техническая эксплуатация и сопровождение информационных систем, квалификация: специалист по технической эксплуатации и сопровождению информационных систем.

Методические рекомендации определяют цели и задачи, конкретное содержание, особенности организации и порядок проведения производственной практики (преддипломной) студентами, а также содержат требования по подготовке отчета о практике.

УДК 33(07)
ББК 65.291р3

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Разработка клиент-серверных приложений с веб-интерфейсом привлекает внимание программистов, администраторов и пользователей тем, что существенно упрощается как сама информационная система организации, так и ее эксплуатация. Наличие центрального сервера позволяет организовать надежное хранение данных и проверку полномочий на доступ к ним. Отсутствие специального программного обеспечения на стороне клиента минимизирует задачи развертывания и администрирования приложений. Для работы клиенту достаточно иметь компьютер с сетевым подключением и стандартный интернет-браузер. Пользователи могут выполнять свои функции с любого компьютера внутренней сети организации или, если открыт доступ, с любого компьютера, подключенного к сети интернет. Это повышает мобильность, позволяет расширить круг пользователей далеко за пределами офиса.

Существуют множество конкурирующих технологий для создания таких систем. Наиболее известные традиционные технологии используют стандартные веб серверы, например, широко распространенные Apache, Zope, IIS и т.д. На некотором поддерживаемом ими языке программирования (Perl, PHP, Python и др.) пишутся программы, генерирующие динамические страницы в ответ на запросы пользователя, в которых широко используются хранимые в СУБД данные. Всё большее внимание разработчиков привлекают Java-технологии, обладающие рядом преимуществ. Наиболее значимыми являются их надежность, обеспечиваемая сертификацией инструментов разработки мировыми лидерами программной индустрии, независимость от платформы, идеология открытого кода и их свободное распространение.

1. ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЕТА ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ (ПРЕДДИПЛОМНОЙ)

Отчет по практической подготовке (далее – отчет) выполняется индивидуально, в соответствии с заданием, применительно к предприятию (организации), утвержденному приказом СПК СКГА как место для прохождения производственной практики (по профилю специальности).

Оформлять текст отчета следует с соблюдением следующих требований:

– набирается шрифтом Times New Roman строчными буквами (заглавные буквы применяются в заголовках 1-го – 2-го уровней);

– поля: верхнее – 2, нижнее – 2, левое – 3, правое – 1;

– отступ первой строки – 1,25 см, оформление абзацев вручную (пробелом) не допускается;

– размер основного шрифта – 14 pt, вспомогательного (подписи к рисункам, сноски и т.д.) – 12 pt;

– интервал междустрочный – одинарный;

– заголовки отделяются от текста сверху на один интервал больше;

– в тексте должен быть установлен автоматический перенос слов;

– между словами не допускается использование более одного пробела, перед знаком препинания пробел не ставится;

– таблицы следует делать в режиме таблиц, все таблицы нумеруют арабскими цифрами в пределах всего текста;

– ширина таблицы должна соответствовать ширине текстового блока отчета;

– формулы должны быть выведены шрифтом такого же размера и начертания, что и основной текст;

– номера формул заключаются в круглые скобки и выравниваются по правому краю печатного листа;

– рисунки, диаграммы, схемы, графики и чертежи, иллюстрации размещаются в тексте после абзацев, содержащих ссылку на них, и должны быть пронумерованы в последовательности, соответствующей упоминанию их в тексте, и номерами привязаны к подрисуночным подписям (точка в конце названия рисунка не ставится);

– ширина рисунков, схем, диаграмм, графиков, чертежей и ил должна соответствовать ширине текстового блока отчета;

– расположение номера страниц – внизу по центру (нумерация страниц на первом листе (титульном) не ставится).

Печатный вариант отчета формируется в скоросшивателе и сдается руководителю практики (от колледжа) вместе с дневником по производственной практике (по профилю специальности).

2. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ И РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ ЗАДАНИЙ НА ПРОИЗВОДСТВЕННУЮ ПРАКТИКУ (ПРЕДДИПЛОМНУЮ)

2.1 Общее ознакомление с ИС предприятия (организации)

Постановка целей и задач практики.

Во введении прописывается:

- информация о целях производственной практики (преддипломной),
- описание перечня задач, которые планируется решить входе прохождения практики в соответствии с рабочей программой,
- указание сроков и места прохождения практической подготовки в соответствии с приказом.

Характеристика предприятия (организации), его подразделений.

Организационная структура предприятия (организации)

В отчет вносятся:

- общие сведения о месте практической подготовки (наименование предприятия (организации), правовой статус, наличие филиалов и представительств),
- краткая характеристика основных видов деятельности по ОКВЭД,

описание организационной структуры управления предприятием (организацией),

- схематичное отображение структурной схемы управления (в виде рисунка).

Характеристика технического, программного и информационного обеспечения предприятия (организации)

Приводится описание всех видов обеспечения внутрипроизводственного обмена информацией: состав и характеристики технического (аппаратного), программного обеспечения, а также информационных систем, используемых на предприятии (организации) или конкретно на месте прохождения практической подготовки, либо обеспечение, предлагаемое в качестве альтернативы имеющемуся, для повышения эффективности выполняемых производственных операций (процессов), с учетом основных видов деятельности предприятия.

Осуществление сбора необходимой информации

Для осуществления поставленных целей и задач практики проводится сбор информации, необходимой для формирования обширной базы данных, являющейся основой будущей информационной системы предприятия (организации). Первичная информация оформляется в отчете в форме таблиц.

2.2 Осуществление интеграции программных модулей

Разработка технического задания

Техническое задание – это документ, определяющий цели, требования и основные исходные данные, необходимые для разработки автоматизированной системы управления. В техническом задании формулируются основные цели разработки, требования к программному продукту, определяются сроки и этапы разработки, регламентируется процесс приемо-сдаточных испытаний.

В разработке технического задания участвуют как представители заказчика, так и исполнитель (обучающийся). В основе этого документа лежат исходные требования заказчика, анализ передовых достижений техники, результаты выполнения научно-исследовательских работ, предпроектных исследований, научного прогнозирования и т.п.

Необходимо разработать техническое задание на создаваемое программное обеспечение (далее – ПО) в соответствии с ГОСТ 19.201-78

«Техническое задание. Требования к содержанию и оформлению». Функциональное назначение программы должно включать в себя описание основных функций программы с представлением входных данных и результатной информации.

Кодирование программного обеспечения

Кодирование – процесс написания программного кода, скриптов, с целью реализации определѐнного алгоритма на определѐнном языке программирования.

Некоторые путают такие понятия, как программирование и непосредственно кодирование. Кодирование является лишь частью программирования, наряду с анализом, проектированием, компиляцией, тестированием и отладкой, сопровождением.

В узких кругах кодирование также может называться «кодинг». Однако в литературе этот термин используется редко.

Стандарт оформления кода (стандарт кодирования, стиль программирования) – набор правил и соглашений, используемых при написании исходного кода на некотором языке программирования. Наличие общего стиля программирования облегчает понимание и поддержание исходного кода, написанного более чем одним программистом, а также облегчает сотрудничество нескольких человек в развитии одного программного обеспечения.

Стандарт оформления кода обычно принимается и используется некоторой группой разработчиков программного обеспечения для единообразного оформления совместно используемого кода. Целью принятия и использования стандарта является упрощение восприятия программного кода человеком, минимизация нагрузки на память и зрение при чтении программы.

В отчете необходимо привести структуру интерфейса разрабатываемой информационной системы в виде форм с их кратким

описанием. Код программного обеспечения может быть размещен в приложении к отчету.

2.3 Платформа Spring

Описание Spring

Платформа Spring один из самых популярных фреймворков для поддержки технологий разработки приложений. Она была разработана в 2003 году Родом Джонсоном. С помощью платформы Spring можно создавать любые приложения на Java.

Платформа Spring основана на использовании важных понятий — внедрение зависимостей (Dependency Injection — DI), что порождает инверсию управления (Inversion of Control — IoC). Данные понятия позволяют при написании крупных и сложных проектов создавать независимые друг от друга классы, пригодные для повторного использования и независимого тестирования. Программная реализация DI обеспечивает связи между экземплярами этих классов, сохраняя их независимость. Все классы приложений на платформе Spring должны удовлетворять требованиям соглашений о бинах (наличие конструк- 9 тора без параметров и свойств, представленных полями для хранения и методами get/set для чтения и записи значений).

Основная идея DI — это внедрение экземпляра бина одного класса в экземпляр другого класса, которое выполняется путем передачи зависимых экземпляров классов через параметры конструктора зависящему классу или с помощью методов инициализации в период исполнения приложения после создания зависящего экземпляра (рис. 1). Бины создаются контейнером на основе объявлений классов и конфигураций приложения. Созданные в нужном количестве бины помещаются в хранилище, откуда они поставляются приложению по запросу на внедрение зависимостей.

Это позволяет в период исполнения не тратить время на создание экземпляров, повышая производительность приложения при обработке потока запросов. Платформа Spring поддерживает технологию аспекто-ориентированного программирования (Aspect oriented programming — AOP).

Данными функциями обслуживаются несколько модулей приложения и независимые от бизнес-логики приложения. В объектно-ориентированном программировании основной единицей является класс, а в АОП основным элементом является аспект. DI позволяет собрать приложение из отдельных классов, а АОП отделяет аспекты от объектов, на поведение которых они влияют.

Сама платформа Spring состоит из отдельных модулей, что позволяет разработчику выбирать, какие из них использовать при разработке приложения (рис. 2). Приведем краткие комментарии состава и функций модулей:. Платформа Spring

Основной контейнер (Core Container) включает в себя модули Beans, Core, Context и SpEL (Spring expression language);

Beans реализует шаблон фабрики классов BeanFactory для создания экземпляров бинов;

модуль Core — центральная часть фреймворка, обеспечивающая функции IoC и DI;

Context построен на основе Beans и Core и позволяет получить доступ к любому бину, который определен в конфигурации приложения, главным элементом модуля Context является интерфейс ApplicationContext;

модуль SpEL поставляет мощный язык Spring-выражений для манипулирования объектами во время исполнения. Рис. 2. Архитектура платформы Spring Контейнер Data Access/Integration содержит драйверы и модули JDBC, ORM, OXM, JMS, Transatcions:

JDBC обеспечивает абстрактный слой драйвера и избавляет разработчика от необходимости вручную прописывать монотонный код для взаимодействия с БД.

ORM обеспечивает интеграцию с технологиями Hibernate, JDO, JPA и т.д.;

модуль OXM поставляет средства работы с файлами формата XML, обеспечивая связь Объект/XML — XMLBeans, JAXB и т.д.

модуль JMS (Java Messaging Service) выполняет создание, передачу и получение сообщений;

модуль Transactions поддерживает управление транзакциями для классов, которые реализуют необходимые методы. Слой Web (MVC-Remoting) состоит из модулей WebSocket, Web, Servlet, Portlet:

Модуль WebSocket обеспечивает поддержку связи между клиентом и сервером, используя Web-Socket для веб-приложений.

Модуль Web выполняет загрузку, передачу файлов и т.д. Web содержит реализацию технологии «модель-вид-контроллер» Spring MVC для веб-приложений.

Servlet — интерфейс Java, расширяющий функциональные возможности сервера в среде портлетов (компонентов пользовательского интерфейса).

Portlet обеспечивает реализацию MVC в среде портлетов. Кроме перечисленных выше, Spring также включает в себя ряд других важных модулей — AOP, Aspects, Instrumentation, Messaging и Test:

AOP реализует аспекто-ориентированное программирование;

модуль Aspects обеспечивает интеграцию с AspectJ, который также

является мощным компонентом АОП;

Instrumentation отвечает за поддержку функций class instrumentation и class loader, которые используются в серверных приложениях;

Messaging обеспечивает поддержку получения и обработки сообщений;

модуль Test обеспечивает тестирование с использованием инструментов TestNG или JUnit. Один из главных принципов поддерживаемой Spring инверсии управления состоит в том, что код программы не должен зависеть от конкретных реализаций классов, а должен зависеть от абстракций, описываемых интерфейсами. Ориентация на интерфейсы — главная черта платформы Spring. Код приложения не должен знать реализацию бина, его функции. Он должен быть уверенным в том, что при запросе бина из хранилища ему будет выдана реализация с запрашиваемым интерфейсом. Это важное отличие от жесткого связывания эк- 12. Платформа Spring земпляров классов в классическом объектно-ориентированном программировании.

Фреймворк Spring-MVC

Фреймворк Spring Web MVC предоставляет архитектуру Model-View-Controller (MVC) и готовые компоненты, которые могут использоваться для разработки гибких и слабо связанных между этими элементами веб-приложений:

модель инкапсулирует данные приложения и состоит из POJO;

представление отвечает за отрисовку данных модели и генерирует выходные страницы HTML, которые может интерпретировать браузер клиента;

контроллер отвечает за обработку запросов пользователей и построение модели, передает ее в представление для отрисовки.

«Словарь данных» (англ. – data dictionary) – это централизованное хранилище метаданных. Оно представляет собой базу данных, созданную для хранения метаданных, т.е. информации о структурах, которые содержат фактические данные.

В отчете приводится разработанный словарь данных, который в табличных формах содержит описание содержащейся информации о названиях полей, форматах и взаимосвязях между ними, сведения о характере использования, а также распределение ответственности.

Фреймворк Spring Boot

Платформа Spring Boot расширяет концепции платформы Spring и позволяет упростить создание автономных вебприложений, готовых для запуска на платформе Java. Платформа Spring Boot и сторонние библиотеки позволяют вести разработку веб-приложений с минимальными усилиями. Большинство приложений Spring Boot не нуждаются в дополнительной конфигурации в виде XML-файлов, что существенно снижает трудоемкость

разработки и повышает качество проекта. Spring Boot наследует от Spring все возможности и концепции с одним большим преимуществом — он создает контейнер в виде файла JAR (его обычно зовут fat jar — толстый jar), в который он внедряет абсолютно все необходимые компоненты, в том числе и сам сервер, для того чтобы приложение могло работать самостоятельно и независимо от программной платформы исполнительного ПК. Например, контейнер может содержать внутри собственный сервер (Tomcat, Netty, Jetty и т.п.), чтобы обслуживать трафик, бизнес-логику и коннекторы для баз данных, брокеров сообщений (Kafka, например) и многое другое. Spring Boot можно использовать для создания приложений Java, которые запускаются из архива jar без традиционного развертывания на целевом сервере из архива war. Основные цели платформы: обеспечить быстрый сценарий разработки; предоставить достаточный инструмент «из коробки» для быстрой разработки проекта.

Управление зависимостями. Каждый выпуск Spring Boot имеет предопределенный список зависимостей, которые он поддерживает. На практике не нужно указывать версию для этих зависимостей в файле конфигурации pom, поскольку Spring Boot управляет этим сам. При обновлении самой Spring Boot данные зависимости будут постоянно обновляться.

Если это необходимо, по-прежнему можете указывать версию и переопределять рекомендации Spring Boot. Список содержит все модули Spring, которые можно использовать, а также уточненный список сторонних библиотек. Список доступен в виде стандартных спецификаций (зависимости), а также доступна дополнительная поддержка Maven и Gradle. Каждый выпуск Spring Boot связан с базовой версией Spring Framework, поэтому настоятельно рекомендуется не указывать его версию самостоятельно. Для сборки проектов Spring Boot использует Maven. При настройке проект необходимо наследовать от стартера spring-boot-starterparent, для чего в файле pom.xml просто прописывается родительский стартер: org.springframework.boot spring-boot-starter-parent X.Y.Z.RELEASE

Если необходимо импортировать дополнительные стартеры, то можно спокойно опускать номер версии. С помощью настройки можно также переопределить отдельные зависимости, задав свойства в собственном проекте. Например, чтобы перейти на другой релиз Spring Data, можно добавить следующий фрагмент в ваш pom.xml.

Фреймворк Spring Boot Fowler-SR2 Управление зависимостями позволяет авторам проекта напрямую указывать версии артефактов, которые будут использоваться, когда они встречаются в зависимостях, являющихся транзитивными, или в зависимостях, в которых версия не указана. Секция управления зависимостями (dependencyManagement) является механизмом централизации информации о зависимостях. Когда у вас есть набор проектов, которые наследуют общий родитель, можно поместить всю информацию о зависимости в общий POM и иметь более простые ссылки на артефакты в дочерних.

Работы с базами данных SQL

Spring Framework предоставляет обширную поддержку для работы с базами данных SQL от прямого доступа JDBC с помощью JdbcTemplate до полной поддержки технологий «Объектно-реляционного отображения», таких как Hibernate. Технология Spring Data обеспечивает дополнительный уровень функциональности, создавая реализации репозитория непосредственно из интерфейсов и используя соглашения для автоматического создания запросов на основе имен методов. Однако следует учитывать многократное замедление работы методов Spring Data по сравнению с простым JDBC, что важно при разработке высоконагруженных приложений.

В целях обеспечения контроля доступа к информации, а также действий и операций, выполняемых пользователями на экземпляре СУБД MS SQL, корпорация Microsoft предлагает штатный инструмент для аудита MS SQL, который предполагает детальное журналирование следующих событий на экземпляре MS SQL.

Детализация и журналирование событий над объектами БД экземпляра СУБД MS SQL:

создание

изменение

удаление.

Детализация и журналирование событий предоставления прав и привилегий к объектам экземпляра СУБД MS SQL.

Детализация и журналирование событий предоставления прав и привилегий к БД экземпляра СУБД MS SQL.

2.4 Проверка данных

Доступны несколько механизмов контроля правильности данных, используемых в различных контекстах приложения: при определении сущностей, в контроллерах для метода POST, на страницах пользовательского интерфейса. Ниже рассмотрен пример, иллюстрирующий приемы верификации данных. Для этого приложения используется язык шаблонов Thymeleaf, поскол Thymeleaf — современный серверный механизм Java-шаблонов для веб, способный обрабатывать HTML, XML, JavaScript, CSS и даже простой текст. Основной целью Thymeleaf является создание удобного способа шаблонизации. Thymeleaf основывается на концепции Natural Templates, чтобы внедрить свою логику в файлы шаблонов таким образом, чтобы шаблон не влиял на отображение дизайна прототипа в браузерах. Thymeleaf разработан с учетом стандартов Web, HTML5, что позволяет создавать соответствующие стандартам шаблоны.

Кроме того, тег будет правильно отображаться всеми браузерами, потому что он не содержит теги, отличные от HTML (браузеры игнорируют все атрибуты, которые они не понимают, например 'th: value'). Но это также позволяет (необязательно) указать в нем атрибут value («Петр Иванов»), который будет отображаться, когда прототип открыт в браузере и будет заменен значением, полученным в результате вычисления выражения ${user.name} во время обработки шаблона

Приложение использует форму с именем класса PersonForm и будет контролировать свойства name и age, для чего создается класс для приема данных с формы при создании новой персоны. Ссылка на бин формы будет нужна при написании шаблона страницы ввода данных от пользователя приложения. Соответствие полей ввода формы и свойств бина формы поддерживается автоматически библиотеками.

После того как вы определили объект для формы, необходимо создать простой веб-контроллер. Контроллер выполняет передачу всех данных из формы, привязанной к объекту PersonForm. В коде получения данных выполняется проверка на наличие ошибок, при обнаружении которых пользователь возвращается с помощью контроллера к исходной странице формы ввода. В этом случае автоматически отображаются все ошибки в атрибутах персоны. Если все атрибуты персоны правильные, браузер перенаправляется на страницу окончательных результатов.

Страница содержит простую форму с полями в отдельных клетках таблицы. Форма предназначена для отправки запроса по адресу корня приложения «/». Он помечается как обработчик объекта формы personForm, который был рассмотрен в методе GET веб-контроллера. Этот объект называется бин-формой. В бине PersonForm есть два свойства, и на странице они использованы с атрибутами th: field="{name}" и th: field="{age}" для

связи со свойствами класса. Рядом с каждым свойством находится дополнительный элемент th: errors, используемый для отображения ошибок результатов проверки. Наконец, есть кнопка для отправки. Как правило, если пользователь вводит имя или возраст, нарушающие ограничения @Valid, он возвращается на эту страницу с отображаемым сообщением об ошибке. После ввода допустимого имени и возраста пользователь перенаправляется контроллером "redirect:/results" на следующую веб-страницу.

2.5 Микросервисная архитектура приложений

Разработка высоконагруженных веб-приложений является комплексной задачей, требующей от программиста знания всего спектра средств информационных технологий. Для такой разработки используют полный стек инструментов, включающий владение техникой создания информационного обеспечения, выбор архитектуры проекта, языки кодирования данных, инструменты и технологии программирования на различных языках, библиотеки и средства поддержки стандартных операций и многое другое. Высоконагруженные приложения требуют высокой скорости подготовки ответа на запросы клиентов. Время реакции сервера складывается из времени на транспортировку запроса от клиента к серверу, подготовку ответа сервером, обратной передачи ответа клиенту и его визуализации. Длина запроса и время на его передачу обычно незначительны.

Основное время тратится на подготовку и передачу ответа клиенту. Традиционные архитектуры веб-приложений на каждый запрос готовят новый HTML-документ, на что затрачивается значительное время, часто определяемое необходимостью поиска информации в базах данных, ее преобразование, форматирование при формировании результирующей страницы и необходимых ей ресурсов. Источником сокращения времени на подготовку ответа является перенос формирования вида страницы на сторону клиента, на сервере остается только формирование необходимых данных.

Эту задачу решает переход на микросервисную архитектуру приложения, когда клиент взаимодействует с набором сервисов, поставляющих данные по запросу клиента, а клиент сам формирует вид страницы для пользователя. Платформа Angular Для разработки сервисов эффективной технологией является RESTful API, реализуемая средствами Spring Boot, — это так называемый 'Back end'. Клиентская часть (Front end) в этом случае реализуется с помощью средств разработки одностраничных приложений (SPA — Single Page Application). Такими платформами являются Angular, React и др.

Платформа Angular

Angular — это фреймворк для создания клиентских приложений. Прежде всего, он нацелен на разработку SPA — одностраничных приложений, содержание которых динамически изменяется при выполнении

манипуляций пользователя органами управления приложением. Для динамического изменения содержимого страниц на стороне браузера используется поддерживаемый всеми браузерами язык JavaScript, библиотеки для манипулирования содержимым страницы, а также объектная модель документа DOM.

Программируя сценарии на языке JavaScript, можно заменять как отдельные элементы страниц, так и всё содержимое сразу. Для управления приложением на странице размещают обычные средства формирования событий, а также создаются функции для их обработки, запрашивающие у сервисов информацию и форматирующие ее для размещения в элементах DOM. Платформа Angular предоставляет именно такую функциональность для управления видом загруженной страницы путем разбиения приложения на компоненты, которые содержат описание программной части в виде классов ООП, шаблонов отображения данных, которые, в свою очередь, содержат вставки выражений с использованием свойств экземпляра класса. Для управления форматами отображения информации широко используются библиотеки тегов, атрибуты тегов и стили.

Angular — это платформа для создания одностраничных клиентских приложений с использованием HTML и TypeScript. Она реализует основные и дополнительные функции в виде набора библиотек TypeScript, которые нужно импортировать в свои приложения. Angular предоставляет различную функциональность для взаимодействия вида с программной частью класса, а именно связывание.

Микросервисная архитектура приложений позволяющее динамически изменять отображаемую информацию при изменении данных, и, наоборот, обработку событий, средства асинхронного взаимодействия с источниками данных, шаблоны, маршрутизацию и многое другое, что составляет функционально полный набор инструментов для разработчика. Одной из ключевых особенностей Angular является то, что платформа использует в качестве языка программирования TypeScript, компилируемый перед исполнением в JavaScript, поддерживаемый всеми браузерами. Поэтому перед началом изучения Angular рекомендуется ознакомиться с основами языка TypeScript, про который можно прочитать на ресурсе https://metanit.com/web/typescript/.

Angular — это фреймворк для построения клиентских приложений на HTML и JavaScript, а также на таком языке, как TypeScript, который компилируется в JavaScript. Фреймворк состоит из нескольких библиотек, часть которых является базовой, а другая часть — факультативной. Создание Angular приложения требует подготовки HTML шаблона с разметкой, написания классов компонентов для управления этими шаблонами, а также реализации логики приложений в службах и объединения компонент и служб в модули. Запуск приложения приводит к загрузке корневого модуля. Angular выполняет управление приложением, демонстрируя его содержимое в браузере и реагируя на взаимодействие с пользователем в соответствии с предоставленными интерфейсами.

2.6 Http и взаимодействие с сервером

HttpClient и отправка запросов

Для взаимодействия с сервером и отправки запросов по протоколу http применяется класс HttpClient. Этот класс определяет ряд методов для отправки различного рода запросов: GET, POST, PUT, DELETE. Данный класс построен поверх стандартного объекта в JavaScript — XMLHttpRequest. Для использования этого класса в проект необходимо установить пакет @angular/common.

При взаимодействии с сервером, как правило, обращения происходят не из самого компонента, а из вспомогательных сервисов, поскольку сервис может определять дополнительную логику обработки полученных с сервера данных, которую мы могли бы выполнить. Кроме того, сервисы могут определять функционал, который будет использоваться несколькими компонентами. Компоненты же выступают в качестве потребителей данных, которые они получают от сервисов.

При работе с сетью и объектом http могут возникать ошибки, например в момент запроса сеть стала недоступна, неправильно указан адрес и соответственно не найден ресурс, либо доступ к ресурсу ограничен, либо другие ошибки. Перехват ошибок позволит выяснить проблему и каким-то образом обработать их, например, вывести пользователю сообщение об ошибке.

Метод subscribe() в качестве второго параметра принимает функцию-обработчик, которая вызывается в случае возникновения ошибки. В этом обработчике можно получить ошибку и обработать ее. Ошибка представляет объект, из которого можно получить ряд данных. В частности, свойство message позволяет получить сообщение об ошибке, а свойство statusCode — статусный код ответа. Кроме того, для перехвата ошибок к объекту Observable, который является результатом запроса, можно применить функцию catchError().

Нередко возникает необходимость обращаться с более сложными запросами к удаленному ресурсу, передавая ему некоторые параметры. Вначале рассмотрим способ передачи параметров в get-запросе. Может использоваться любая технология на сервере сервисов. Но вне зависимости от выбранной технологии следует учитывать ограничения на кросс-доменные запросы. В частности, если наше приложение на Angular запускается в одном домене, а сервер, обрабатывающий запросы, запущен на другом домене, то на сервере нам надо включить технологию CORS. Поскольку данные передаются через запрос GET, то мы можем конкатенировать нужное число в строке запроса.

3. ЗАДАНИЕ НА ПРОИЗВОДСТВЕННУЮ ПРАКТИКУ (ПРЕДДИПЛОМНУЮ)

Практическая работа № 1

Задание 1 Для оценки сформированности компетенций ОК 01.; ОК 02.; ОК 03.; ОК 04.; ОК 05.; ОК 06.; ОК 07.; ОК 08.; ОК 09.; ПК 1.1.; ПК 1.2.; ПК 1.3.; ПК 1.4.; ПК 1.5.; ПК 1.6.; ПК 1.7.; ПК 2.1.; ПК 2.2.; ПК 2.3.; ПК 2.4.; ПК 2.5.; ПК 2.6.
Задание 1. Создание простой веб-страницы с заголовком, основным текстом, панелью навигации.

Задание 2. Добавление стилей CSS для редактирования внешнего вида страницы и ее элементов.

Критерии оценки:
Устанавливаются с учетом балльно-рейтинговой системы по дисциплине и выражаются в баллах.
Максимальная оценка за задание 6 баллов, минимальная – 5 баллов.

Критерии оценки	
6	Задания выполнены в полном объеме. Оформление и результаты соответствуют требованиям. Выводы сформулированы
5,5	Задания выполнены в полном объеме имеются неточности в оформлении, имеются неточности в формулировке выводов.
5	Задания выполнены не в полном объеме, имеются неточности в оформлении, имеются неточности в формулировке выводов.

Практическая работа № 2

(для оценки сформированности компетенции ОК 01.; ОК 02.; ОК 03.; ОК 04.; ОК 05.; ОК 06.; ОК 07.; ОК 08.; ОК 09.; ПК 1.1.; ПК 1.2.; ПК 1.3.; ПК 1.4.; ПК 1.5.; ПК 1.6.; ПК 1.7.; ПК 2.1.; ПК 2.2.; ПК 2.3.; ПК 2.4.; ПК 2.5.; ПК 2.6.)

Задание 1. Написание кода на JavaScript для вывода текста в консоль.

Задание 2. Написание кода на JavaScript для работы с простыми числами, их сложение, вычитание, деление, умножение, возведение в степень.

Задание 3. Создание веб-страницы с добавлением функций на JavaScript.

Критерии оценки:
Устанавливаются с учетом балльно-рейтинговой системы по дисциплине и

выражаются в баллах.
Максимальная оценка за задание 3 балла, минимальная – 2 балла.

Критерии оценки	
3	Задания выполнены в полном объеме. Оформление и результаты соответствуют требованиям. Выводы сформулированы
2,5	Задания выполнены в полном объеме имеются неточности в оформлении, имеются неточности в формулировке выводов.
4	Задания выполнены не в полном объеме, имеются неточности в оформлении, имеются неточности в формулировке выводов.

Практическая работа № 3

(для оценки сформированности компетенций ОК 01.; ОК 02.; ОК 03.; ОК 04.; ОК 05.; ОК 06.; ОК 07.; ОК 08.; ОК 09.; ПК 1.1.; ПК 1.2.; ПК 1.3.; ПК 1.4.; ПК 1.5.; ПК 1.6.; ПК 1.7.; ПК 2.1.; ПК 2.2.; ПК 2.3.; ПК 2.4.; ПК 2.5.; ПК 2.6.)

Задание 1. Написание кода на JavaScript, который обрабатывает действие кнопок мыши и в соответствии с нажатием выводит текст с номером кнопки.

Задание 2. Создание веб-страницы и добавление на нее объектов для работы с целыми числами (кнопка, поле ввода, поле вывода).

Задание 3. Создание веб-страницы с различными кнопками, которые выполняют различные действия (изменение цвета страницы, добавление новых объектов)

Задание 4. Создание веб-страницы с изображением, которое при клике мышью меняет свой размер.

Задание 5. Создание веб-страницы с изображением, которое при наведение курсором становится прозрачным.

Критерии оценки:
Устанавливаются с учетом балльно-рейтинговой системы по дисциплине и выражаются в баллах.
Максимальная оценка за задание 3 балла, минимальная – 2 балла.

Критерии оценки	
3	Задания выполнены в полном объеме. Оформление и результаты соответствуют требованиям. Выводы сформулированы

2,5	Задания выполнены в полном объеме имеются неточности в оформлении, имеются неточности в формулировке выводов.
2	Задания выполнены не в полном объеме, имеются неточности в оформлении, имеются неточности в формулировке выводов.

Практическая работа № 4

Задание 1 (для оценки сформированности компетенции ОК 01.; ОК 02.; ОК 03.; ОК 04.; ОК 05.; ОК 06.; ОК 07.; ОК 08.; ОК 09.; ПК 1.1.; ПК 1.2.; ПК 1.3.; ПК 1.4.; ПК 1.5.; ПК 1.6.; ПК 1.7.; ПК 2.1.; ПК 2.2.; ПК 2.3.; ПК 2.4.; ПК 2.5.; ПК 2.6.).
Задание 1. Написание кода на JavaScript, который находит элемент DOM с использованием его идентификатора.

Задание 2. Написание кода на JavaScript, который меняет текст элемента DOM.

Задание 3. Создание веб-страницы на котором располагаются кнопка и поле ввода, при нажатии на кнопку в поле ввода появляется любое целое число.

Критерии оценки:
Устанавливаются с учетом балльно-рейтинговой системы по дисциплине и выражаются в баллах.
Максимальная оценка за задание 3 балла, минимальная – 2 балла.

Критерии оценки	
3	Задания выполнены в полном объеме. Оформление и результаты соответствуют требованиям. Выводы сформулированы
2,5	Задания выполнены в полном объеме имеются неточности в оформлении, имеются неточности в формулировке выводов.
2	Задания выполнены не в полном объеме, имеются неточности в оформлении, имеются неточности в формулировке выводов.

Практическая работа № 5

Задание 1 (для оценки сформированности компетенции ОК 01.; ОК 02.; ОК 03.; ОК 04.; ОК 05.; ОК 06.; ОК 07.; ОК 08.; ОК 09.; ПК 1.1.; ПК 1.2.; ПК 1.3.; ПК 1.4.; ПК 1.5.; ПК 1.6.; ПК 1.7.; ПК 2.1.; ПК 2.2.; ПК 2.3.; ПК 2.4.; ПК 2.5.; ПК 2.6.).
Проведение самоанализа разработанных ранее веб-сайтов, анализ соответствия заданий и итоговых результатов.

Критерии оценки:
Устанавливаются с учетом балльно-рейтинговой системы по дисциплине и выражаются в баллах.
Максимальная оценка за задание 3 балла, минимальная – 2 балла.

Критерии оценки	
3	Задания выполнены в полном объеме. Оформление и результаты соответствуют требованиям. Выводы сформулированы
2,5	Задания выполнены в полном объеме имеются неточности в оформлении, имеются неточности в формулировке выводов.
2	Задания выполнены не в полном объеме, имеются неточности в оформлении, имеются неточности в формулировке выводов.

Практическая работа № 6

Задание 1 (для оценки сформированности компетенции ОК 01.; ОК 02.; ОК 03.; ОК 04.; ОК 05.; ОК 06.; ОК 07.; ОК 08.; ОК 09.; ПК 1.1.; ПК 1.2.; ПК 1.3.; ПК 1.4.; ПК 1.5.; ПК 1.6.; ПК 1.7.; ПК 2.1.; ПК 2.2.; ПК 2.3.; ПК 2.4.; ПК 2.5.; ПК 2.6.).
Проведение анализа существующих сайтов, формирование общей насмотренности и понимания тенденций в области веб-разрботки.

Критерии оценки:
Устанавливаются с учетом бально-рейтинговой системы по дисциплине и выражаются в баллах.
Максимальная оценка за задание 3 балла, минимальная – 2 балла.

Критерии оценки	
3	Задания выполнены в полном объеме. Оформление и результаты соответствуют требованиям. Выводы сформулированы
2,5	Задания выполнены в полном объеме имеются неточности в оформлении, имеются неточности в формулировке выводов.
2	Задания выполнены не в полном объеме, имеются неточности в оформлении, имеются неточности в формулировке выводов.

Практическая работа № 7

(для оценки сформированности компетенции ОК 01.; ОК 02.; ОК 03.; ОК 04.; ОК 05.; ОК 06.; ОК 07.; ОК 08.; ОК 09.; ПК 1.1.; ПК 1.2.; ПК 1.3.; ПК 1.4.; ПК 1.5.; ПК 1.6.; ПК 1.7.; ПК 2.1.; ПК 2.2.; ПК 2.3.; ПК 2.4.; ПК 2.5.; ПК 2.6.)

Задание 1. Разработка веб-сайта с использованием асинхронного JavaScript;

Задание 2. Реализация функции, выполняющейся через предопределенный интервал времени;

Задание 3. Реализация на веб-странице таймера обратного отсчета, отображающего оставшееся время до конкретной даты;

Задание 4. Реализация загрузки данных с JSON с сервера и отображает их в таблице.

Критерии оценки:
Устанавливаются с учетом балльно-рейтинговой системы по дисциплине и выражаются в баллах.
Максимальная оценка за задание 2 балла, минимальная – 1 балл.

Критерии оценки	
2	Задания выполнены в полном объеме. Оформление и результаты соответствуют требованиям. Выводы сформулированы
1,5	Задания выполнены в полном объеме имеются неточности в оформлении, имеются неточности в формулировке выводов.
1	Задания выполнены не в полном объеме, имеются неточности в оформлении, имеются неточности в формулировке выводов.

Практическая работа № 8

(для оценки сформированности компетенции ОК 01.; ОК 02.; ОК 03.; ОК 04.; ОК 05.; ОК 06.; ОК 07.; ОК 08.; ОК 09.; ПК 1.1.; ПК 1.2.; ПК 1.3.; ПК 1.4.; ПК 1.5.; ПК 1.6.; ПК 1.7.; ПК 2.1.; ПК 2.2.; ПК 2.3.; ПК 2.4.; ПК 2.5.; ПК 2.6.).

Задание 1. Интеграция библиотеки jQuery в разрабатываемый веб-сайт

Критерии оценки:
Устанавливаются с учетом балльно-рейтинговой системы по дисциплине и выражаются в баллах.

Максимальная оценка за задание 3 балла, минимальная – 2 балла.

Критерии оценки	
3	Задания выполнены в полном объеме. Оформление и результаты соответствуют требованиям. Выводы сформулированы
2,5	Задания выполнены в полном объеме имеются неточности в оформлении, имеются неточности в формулировке выводов.
2	Задания выполнены не в полном объеме, имеются неточности в оформлении, имеются неточности в формулировке выводов.

Практическая работа № 9

Задание 1 (для оценки сформированности компетенции ОК 01.; ОК 02.; ОК 03.; ОК 04.; ОК 05.; ОК 06.; ОК 07.; ОК 08.; ОК 09.; ПК 1.1.; ПК 1.2.; ПК 1.3.; ПК 1.4.; ПК 1.5.; ПК 1.6.; ПК 1.7.; ПК 2.1.; ПК 2.2.; ПК 2.3.; ПК 2.4.; ПК 2.5.; ПК 2.6.).

Разработка веб-сайта с использованием php для интеграции сайта и базы данных.

Критерии оценки:
Устанавливаются с учетом балльно-рейтинговой системы по дисциплине и выражаются в баллах.
Максимальная оценка за задание 3 балла, минимальная – 2 балла.

Критерии оценки	
3	Задания выполнены в полном объеме. Оформление и результаты соответствуют требованиям. Выводы сформулированы
2,5	Задания выполнены в полном объеме имеются неточности в оформлении, имеются неточности в формулировке выводов.
2	Задания выполнены не в полном объеме, имеются неточности в оформлении, имеются неточности в формулировке выводов.

Практическая работа № 10

Задание 1 (для оценки сформированности компетенции ОК 01.; ОК 02.; ОК 03.; ОК 04.; ОК 05.; ОК 06.; ОК 07.; ОК 08.; ОК 09.; ПК 1.1.; ПК 1.2.; ПК 1.3.; ПК 1.4.; ПК 1.5.; ПК 1.6.; ПК 1.7.; ПК 2.1.; ПК 2.2.; ПК 2.3.; ПК 2.4.; ПК 2.5.;

ПК 2.6.).

Разработка веб-сайта с использованием инструментов Joomla.

Критерии оценки:
Устанавливаются с учетом балльно-рейтинговой системы по дисциплине и выражаются в баллах.
Максимальная оценка за задание 3 балла, минимальная – 2 балла.

Критерии оценки	
3	Задания выполнены в полном объеме. Оформление и результаты соответствуют требованиям. Выводы сформулированы
2,5	Задания выполнены в полном объеме имеются неточности в оформлении, имеются неточности в формулировке выводов.
2	Задания выполнены не в полном объеме, имеются неточности в оформлении, имеются неточности в формулировке выводов.

Практическая работа № 11

Задание 1 (для оценки сформированности компетенции ОК 01.; ОК 02.; ОК 03.; ОК 04.; ОК 05.; ОК 06.; ОК 07.; ОК 08.; ОК 09.; ПК 1.1.; ПК 1.2.; ПК 1.3.; ПК 1.4.; ПК 1.5.; ПК 1.6.; ПК 1.7.; ПК 2.1.; ПК 2.2.; ПК 2.3.; ПК 2.4.; ПК 2.5.; ПК 2.6.).

Подготовка и сдача отчетных материалов.
Обсуждение и анализ результатов практики.

Критерии оценки:
Устанавливаются с учетом балльно-рейтинговой системы по учебной практике и выражаются в баллах.

1. Выставление оценок осуществляется на основе принципов объективности, справедливости, всестороннего анализа уровня знаний обучающихся.
2. При выставлении оценки преподаватель учитывает:

- знание фактического материала по программе, в том числе знание обязательной литературы, современных публикаций по программе курса, а также истории науки;
- степень выполнения заданий текущего контроля;
- логику, структуру, стиль ответа; культуру речи, манеру общения; готовность к дискуссии, аргументированность ответа; уровень самостоятельного мышления; умение приложить теорию к практике, решить задачи;

3. Оценка «отлично» (30 баллов).

Оценка «отлично» ставится обучающемуся, ответ которого содержит:

- глубокое знание теоретического материала в соответствии с элементами формируемых дисциплиной компетенций, а также основного содержания и новаций лекционного курса по сравнению с учебной литературой;
- знание концептуально-понятийного аппарата всего курса.

А также свидетельствует о способности:

- самостоятельно критически оценивать основные положения курса;
- увязывать теорию с практикой.

Оценка «отлично» не ставится в случаях систематических пропусков обучающимся аудиторных занятий по неуважительным причинам, отсутствия активного участия на практических занятиях, а также неправильных ответов на дополнительные вопросы преподавателя.

Оценка «хорошо» (20 баллов).

Оценка «хорошо» ставится обучающемуся, ответ которого свидетельствует:

- о полном знании материала по программе в соответствии с элементами формируемых дисциплиной компетенций;
- о знании рекомендованной литературы;
- содержит в целом правильное, но не всегда точное и аргументированное изложение материала.

Оценка «хорошо» не ставится в случаях пропусков обучающимся аудиторных занятий по неуважительным причинам.

Оценка «удовлетворительно» (10 баллов) ставится обучающемуся, ответ которого содержит:

- поверхностные знания важнейших разделов программы и содержания лекционного курса в соответствии с элементами формируемых дисциплиной компетенций;
- затруднения с использованием научно-понятийного аппарата и терминологии курса;
- стремление логически четко построить ответ, а также свидетельствует о возможности последующего обучения.

Оценки «неудовлетворительно» (0 баллов).

- Оценки «неудовлетворительно» ставятся обучающемуся, имеющему существенные пробелы в знании основного материала по программе, а также допустившему принципиальные ошибки при изложении материала.

4. СПИСОК РЕКОМЕНДУЕМЫХ ИСТОЧНИКОВ

1. Фуфаев, Д.Э. Разработка и эксплуатация автоматизированных информационных систем [Текст]: учебник для студ. учреждений сред. проф. образования / Д.Э. Фуфаев, Э.В. Фуфаев. - М.: Академия, 2017.- 304с.
2. Николаев Е.И. Базы данных в высокопроизводительных информационных системах [Электронный ресурс] : учебное пособие / Е.И. Николаев. — Электрон. текстовые данные. — Ставрополь: Северо-Кавказский федеральный университет, 2016. — 163 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/69375.html
3. Анкудинов И.Г. Информационные системы и технологии [Электронный ресурс] : учебник / И.Г. Анкудинов, И.В. Иванова, Е.Б. Мазаков. — Электрон. текстовые данные. — СПб. : Национальный минерально-сырьевой университет «Горный», 2017. — 259 c. — 978-5-94211- 729-0. — Режим доступа: http://www.iprbookshop.ru/71695.html Электронные издания (электронные ресурсы)
4. Долженко, А. И. Технологии командной разработки программного обеспечения информационных систем: курс лекций / А. И. Долженко. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Эр Медиа, 2019. — 300 c. — ISBN 978-5-4486-0525-3. —
Текст: электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: https://www.iprbookshop.ru/79723.html. — Режим доступа: для авторизир. пользователей
5. Грекул, В. И. Управление внедрением информационных систем : учебное пособие для СПО / В. И. Грекул, Г. Н. Денищенко, Н. Л. Коровкина.
— Саратов: Профобразование, 2021. — 277 c. — ISBN 978-5-4488-1016-9. — Текст: электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: https://www.iprbookshop.ru/102209.html. — Режим доступа: для авторизир. пользователей
6. Долженко, А. И. Управление информационными системами : учебное пособие / А. И. Долженко. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 180 c. — ISBN 978-5-4497-0911-0. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: https://www.iprbookshop.ru/102074.html. — Режим доступа: для авторизир. пользователей
7. Грекул В.И. Проектирование информационных систем. Курс лекций [Электронный ресурс] : учебное пособие для студентов вузов, обучающихся по специальностям в области информационных технологий / В.И. Грекул, Г.Н. Денищенко, Н.Л. Коровкина. — Электрон. текстовые данные. — Москва, Саратов: ИнтернетУниверситет Информационных Технологий (ИНТУИТ),

Вузовское образование, 2017. — 303 c. — 978-5-4487-0089-7. — Режим доступа: http://www.iprbookshop.ru/67376.html

8. Федорова, Г.Н. Информационные системы [Текст]: учебник для студ. учреждений сред проф. образования / Г.Н. Федорова.- М.: Академия, 2017.- 208 с.