ООО «АКАДЕМИЯ ЛАД»

УТВЕРЖДЕНО
Директором ООО «Академия Лад»

_____А.В. Усков

(Приказ №1 от 23 мая 2023 г.)

ДОПОЛНИТЕЛЬНОЕ ОБРАЗОВАНИЕ

ДОПОЛНИТЕЛЬНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАНИЕ

ДОПОЛНИТЕЛЬНАЯ ПРОФЕССИОНАЛЬНАЯ ПРОГРАММА - ПРОГРАММА ПРОФЕССИОНАЛЬНОЙ ПЕРЕПОДГОТОВКИ

«Fullstack-разработка на JavaScript»

(Наименование программы)

288 часа

Нижний Новгород

2023

1. ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

1.1. Цель программы: совершенствование и получение новой компетенции, профессиональных знаний, которые позволят реализовать себя в сфере fullstack-разработки на JavaScript, React и Node.js.

1.2. Нормативные документы для разработки программы профессиональной переподготовки:

- Федеральный закон от 29.12.2012 № 273-ФЗ «Об образовании в Российской Федерации»;

- Приказ Министерства образования и науки РФ от 1 июля 2013 г. № 499 «Об утверждении Порядка организации и осуществления образовательной деятельности по дополнительным профессиональным программам»;

- Профессиональный стандарт 06.001 «Программист» утвержденный приказом Министерства труда и социальной защиты Российской Федерации от 20.07.2022 № 424н.

1.3. Категории слушателей на обучение которых рассчитана программа профессиональной переподготовки (далее – Программа): лица, имеющие среднее профессиональное или высшее образование.

1.4. Входные требования к обучающимся:

Наличие базовых навыков работы с персональным компьютером.

1.5. Сфера применения слушателями полученных профессиональных компетенций, умений и знаний. Знания, полученные в ходе программы могут быть использованы на предприятиях малого и среднего бизнеса, в частной практике при реализации деятельности по fullstack-разработке.

1.6. Программа реализуется исключительно с применением электронного обучения и дистанционных образовательных технологий.

2. ХАРАКТЕРИСТИКА ПОДГОТОВКИ ПО ПРОГРАММЕ

2.1. Нормативный срок освоения программы 288 часов.

2.2. Срок обучения 40 недель

2.3. Общая трудоемкость 8 ЗЕ.

2.4. Режим обучения 7,2 часа в неделю.

3. ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОБУЧЕНИЯ

Слушатель, освоивший программу, должен:

3.1. обладать профессиональными компетенциями, включающими в себя способность:

ПК-1. Техническая поддержка процессов, создание (модификация) и анализ и сопровождение информационных ресурсов.

ПК-2. Выполнение работ по созданию (модификации) и сопровождению информационных ресурсов.

ПК-3. управлять процессами и проектами по созданию (модификации) информационных ресурсов.

3.2. знать:

- основы веб-программирования на языке JavaScript;
- основные технологии разработки сайтов (flexbox, css grid)
- основы разработки веб-приложений на React;
- инструменты разработки веб-приложений;
- основы backend-разработки на Node.js и Nest.js;
- основы реляционных баз данных и языка SQL;
- основы работы с GNU/Linux, Nginx и Docker;
- инструменты backend-разработки веб-приложений;
- методы и приемы отладки программного кода.

3.3. уметь:

- создавать сайты с фиксированной шириной, гибкой вёрсткой и адаптировать сайты под различные размеры экрана;
- выявлять ошибки в программном коде;
- разрабатывать frotnend сайтов и приложения на языке JavaScript, в том числе с использованием библиотек, инструментов и фреймворков (React, React Redux);
- разрабатывать backend сайтов на языке JavaScript, в том числе с использованием библиотек, инструментов и фреймворков (Node.js и Nest.js);
- проектировать базы данных;
- создавать проект backend веб-приложений на NestJS и работать с миграциями TypeORM;
- настраивать веб-сервер Nginx, работать с Docker и Docker-Compose.
- работать в команде.

3.4. владеть:

- современными информационными технологиями и навыками работы с программным обеспечением для разработки веб-приложений;
- технологиями разработки веб-приложений с использованием языка JavaScript и фреймворка React;
- технологиями backend-разработки веб-приложений с использованием языка JavaScript и инструментов Node.js и Nest.js;
- навыками работы с GNU/Linux, Nginx и Docker.

3.5. Сфера применения слушателями полученных профессиональных компетенций, умений и знаний.

Знания, полученные в ходе программы могут быть использованы на предприятиях малого и среднего бизнеса, в частной практике при реализации деятельности по fullstack-разработке веб-приложений.

4. ТРЕБОВАНИЯ К СТРУКТУРЕ ПРОГРАММЫ

Программа предусматривает изучение следующих дисциплин:

Дисциплина 1. «Вёрстка сайтов»

Дисциплина 2. «Программирование на JavaScript»

Дисциплина 3. «Frontend-разработка на React»

Дисциплина 4. «Основы backend-разработки на Node.js»

Дисциплина 5. «Базы данных»

Дисциплина 6. «Фреймворк NestJS»

Дисциплина 7. «GNU/Linux, Nginx, Docker»

5. ТРЕБОВАНИЯ К ОЦЕНКЕ КАЧЕСТВА ОСВОЕНИЯ ПРОГРАММЫ ПРОФЕССИОНАЛЬНОЙ ПЕРЕПОДГОТОВКИ

«Fullstack-разработка на JavaScript»

Процедура **промежуточной аттестации** предусматривает выполнение практических заданий по темам дисциплин.

Процедура **итоговой аттестации** предусматривает выполнение итогового проекта в команде. При подготовке итогового проекта слушатель должен продемонстрировать полученные умения и навыки в области fullstack-разработки веб-приложений. Текущий контроль проверки качества освоения дисциплин программы профессиональной подготовки осуществляется в форме устного собеседования.

Таблица 1

Формы и методы контроля и оценки результатов освоения дисциплин

№ п/п	Наименование процедуры	Основные показатели оценки	Формы и методы контроля и оценки
1.	Дисциплина 1. «Вёрстка сайтов»	*знает* основные инструменты и технологии разработки веб-приложений (flexbox, css grid), методы и приемы отладки программного кода	Текущий контроль - устный опрос. Промежуточная аттестация - выполнение

		умеет создавать сайты с фиксированной шириной, гибкой вёрсткой и адаптировать сайты под различные размеры экрана, выявлять ошибки в программном коде, работать в команде *владеет* современными информационными технологиями и навыками работы со специальным программным обеспечением для разработки веб-приложений; технологиями разработки веб-приложений.	практических заданий.
2.	Дисциплина 2. «Программирование на JavaScript»	*знает* основы веб-программирования на языке JavaScript, методы и приемы отладки программного кода *умеет* решать задачи программирования на JavaScript. *владеет* современными информационными технологиями и навыками программирования на языке JavaScript	Текущий контроль - устный опрос. Промежуточная аттестация - выполнение практических заданий.
3.	Дисциплина 3. «Frontend-разработка на React»	*знает* основы разработки веб-приложений на React, инструменты разработки веб-приложений, методы и приемы отладки программного кода *умеет* разрабатывать сайты и приложения на языке JavaScript, в том числе с использованием библиотек, инструментов и фреймворков (React, React Redux), работать в команде *владеет* современными информационными технологиями и навыками работы со специальным программным обеспечением для разработки веб-приложений; технологиями разработки веб-приложений с использованием языка JavaScript и фреймворка React	Текущий контроль - устный опрос. Промежуточная аттестация - выполнение практических заданий.

4.	Дисциплина 4. «Основы backend-разработки на Node.js»	*знает* основы работы с Node.js. *умеет* создать простой сервер на Node.js., CRUD API, приложение на фреймворке Express *владеет* современными информационными технологиями и навыками работы с платформой с Node.js.	Текущий контроль - устный опрос. Промежуточная аттестация - выполнение практических заданий.
5.	Дисциплина 5. «Базы данных»	*знает* основы реляционных баз данных и языка SQL *умеет* проектировать базы данных (DDL инструкции языка SQL для управления структурой БД и DML инструкции языка SQL для управления данными) *владеет* навыками работы с программным обеспечением для backend-разработки веб-приложений.	Текущий контроль - устный опрос. Промежуточная аттестация - выполнение практических заданий.
6.	Дисциплина 6. «Фреймворк NestJS»	*знает* основы TypeScript, CLI-инструментарий фреймворка NestJS. *умеет* создавать проект backend веб-приложений на NestJS и собирать и запускать готовое приложение. *владеет* инструментарием объектро-реляционного отображения TypeORM и понимает концепцию работы миграций.	Текущий контроль - устный опрос. Промежуточная аттестация - выполнение практических заданий.
7.	Дисциплина 7. «GNU/Linux, Nginx, Docker»	*знает* основы работы в интерпретаторе командной строки GNU/Linux, *умеет* устанавливать программное обеспечение из репозитория GNU/Linux (Docker, Nginx), настраивать веб-сервер Nginx, работать с Docker и Docker-Compose. *владеет* навыками работы с GNU/Linux, Nginx и Docker.	Текущий контроль - устный опрос. Промежуточная аттестация - выполнение практических заданий.
8.	Итоговая аттестация	Работа в качестве Fullstack-разработчика на стажировке.	Итоговый проект

Примеры вопросов для текущего контроля освоения учебного материала:

1. Чем отличается тег от тега в HTML? Какие задачи они обычно выполняют?
2. Каким образом можно создать адаптивный (responsive) дизайн веб-страницы с использованием CSS? Опишите основные подходы.
3. Что такое селекторы в CSS? Приведите примеры различных типов селекторов и их применение.
4. Каким образом можно центрировать элементы на веб-странице с помощью CSS? Объясните различные методы.
5. Что такое блочная модель в CSS? Какие атрибуты она включает в
6. Как объявить переменную в JavaScript с использованием ключевых слов let, const и var? В чем различия между ними?
7. Что такое функции обратного вызова (callback functions) в JavaScript? Как они используются в асинхронном программировании?
8. Каким образом можно выполнить итерацию по элементам массива в JavaScript? Приведите примеры использования циклов и методов массивов.
9. Что такое объектно-ориентированное программирование (ООП) в контексте JavaScript? Как создать класс и экземпляр класса?
10. Каким образом можно обрабатывать ошибки (exceptions) в JavaScript с помощью конструкции try-catch? Приведите примеры использования.
11. Что такое JSX в React? В чем преимущества его использования по сравнению с обычным JavaScript?
12. Каким образом можно передавать данные от родительского компонента к дочернему в React? Опишите механизм использования props.

13. Что такое состояние (state) компонента в React? Как оно устанавливается и изменяется?
14. Каким образом можно организовать обработку событий (event handling) в React? Приведите примеры использования.
15. Что такое жизненный цикл компонента в React? Опишите основные методы жизненного цикла и их назначение.
16. Каким образом можно организовать маршрутизацию (routing) в React-приложении? Какие библиотеки или подходы можно использовать для этого?
17. В чем заключается концепция виртуального DOM (virtual DOM) в React? Какие преимущества она предоставляет при разработке интерфейсов?
18. Каким образом можно выполнять запросы к серверу (AJAX) в React-приложении? Опишите основные подходы и библиотеки.
19. Что такое контекст (context) в React? Как он используется для передачи данных через дерево компонентов?
20. Каким образом можно оптимизировать производительность React-приложения? Укажите основные методы оптимизации.
21. Что такое Node.js и для чего он используется?
22. Какие основные модули входят в стандартную библиотеку Node.js?
23. Каковы преимущества асинхронной модели в Node.js?
24. Какие типы баз данных существуют, и в чем их отличия?
25. Что такое SQL и NoSQL базы данных? Приведите примеры каждого типа.
26. Какие операции можно выполнять с помощью языка SQL?
27. Какие типы данных поддерживает PostgreSQL?
28. Что такое транзакции в PostgreSQL и почему они важны?
29. Как создать индекс в PostgreSQL и в чем заключается его практическая польза?

30. Как установить и настроить соединение с базой данных PostgreSQL в приложении Node.js?
31. Какие библиотеки или ORM вы использовали для работы с PostgreSQL из Node.js, и почему выбрали именно их?
32. Как вы обрабатываете ошибки при выполнении запросов к базе данных из приложения на Node.js?
33. Чем отличается NestJS от других Node.js фреймворков?
34. Какие основные концепции используются в NestJS?
35. Как организовать обработку HTTP-запросов с помощью NestJS?
36. Что такое GNU/Linux, и в чем его основные отличия от других операционных систем?
37. Какие основные команды используются в командной строке Linux?
38. Как создать пользовательский процесс в Linux?
39. Зачем используется веб-сервер Nginx?
40. Как настроить проксирование запросов с помощью Nginx?
41. Какие основные преимущества Nginx перед другими веб-серверами?
42. Что такое Docker и какие преимущества он предоставляет для разработки и деплоя приложений?
43. Как создать Docker-контейнер для приложения на Node.js?
44. Какие основные команды Docker используются для управления контейнерами?

Примеры практических заданий для осуществления промежуточной аттестации:

Дисциплина 1. «Вёрстка сайтов»

1. В файле index.html описать структуру страницы в тэгах. Ссылка на макет https://www.figma.com/file/Itn0zsUrZWQXCgNuPSpxIS/%D1%81%D0

%B8%D0%BD%D0%B8%D0%B9-%D0%BB%D1%83%D0%B3?node-id=0%3A1&t=iaICgm9EJa7GTndx-0. Верстать не надо Создавать style.css не надо.

2. Для проекта в задании 1 создать style.css. Применить свойства, которые проходили на занятии (размеры и цвет шрифта, ширина и высота блоков).
3. Сверстать макет https://www.figma.com/file/Itn0zsUrZWQXCgNuPSpxIS/%D1%81%D0%B8%D0%BD%D0%B8%D0%B9-%D0%BB%D1%83%D0%B3?node-id=0%3A1&t=iaICgm9EJa7GTndx-0 на flexbox и CSS Grid.
4. Описать структуру страницы согласно БЭМ. Верстка страницы на flex и grid с переиспользованием классов и с учетом codeguide. Ссылка на макет https://www.figma.com/file/NyOokSAieFmcsBK3rW0wwN/%D0%94%D0%BE%D1%81%D0%BA%D0%B0-%D0%BE%D0%B1%D1%8A%D1%8F%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B9?node-id=0-1&t=rXzoWsah9vg09rVj-0 .

Дисциплина 2. «Программирование на JavaScript»

1. Придумать наименования переменных для следующих примеров:

- Переменная для “названия нашей планеты”?
- Переменная для “текущее время пользователя”?
- Переменная которая показывает “количества статей”?
- Переменная которая показывает “это оплата наличными деньгами или нет”?
- Три переменные для хранения Ф.И.О

2. Решить задачи с использованием циклов while и for.

- Вывести в консоль заданную строку N раз.

- Ежедневно количество доступных автомобилей в салоне уменьшается в два раза. Выяснить, на какой день продаж, количество доступных к покупке авто станет меньше M, если известно, что в первый день продаж всего было N автомобилей.
- Проанализировав временной промежуток начиная с 1800 и до 2022 года найти и вывести в консоль: Год первого полета человека в космос и количество итераций которое потребовалось для поиска. Количество високосных годов принадлежащих данному отрезку и количество итераций которое потребовалось для поиска.

3. Стрелочные функции

- Напишите стрелочную функцию которая будет выводить переданную строку в консоль n раз.
- Напишите стрелочную функцию, которая будет принимать в качестве параметра букву и если она гласная, функция будет возвращать true, в противном случае false .
- Напишите стрелочную функцию, которая будет возвращать true если строка является палиндромом и false в противном случае.

4. Функции

- Реализовать функцию которая будет принимать числовой диапазон в качестве параметров [min, max] и будет возвращать случайное целое число из данного диапазона.
- Реализовать функцию которая будет определять, в каком регистре записан n элемент переданной строки, если в верхнем то вернуть true, в противном случае вернуть false.
- Реализовать функцию которая заменяет в строке str, все вхождения подстроки find, на подстроку replace.

Дисциплина 3. «Frontend-разработка на React»

1. Создание компонента "Список задач" (To-Do List):
   - Создайте компонент для отображения списка задач.
   - Реализуйте функционал добавления новой задачи, удаления задачи и отметки задачи как выполненной.
   - Добавьте возможность фильтрации задач по статусу (выполнено/не выполнено).
2. Разработка простого приложения "Калькулятор":
   - Создайте компонент калькулятора, который позволяет выполнять базовые математические операции (сложение, вычитание, умножение, деление).
   - Добавьте функционал для очистки экрана и выполнения вычислений.
3. Интеграция с API для отображения данных:
   - Используйте API для получения данных (например, список пользователей, постов или другую информацию).
   - Создайте компоненты для отображения этих данных в удобном формате (список, карточки и т. д.).
4. Форма обратной связи:
   - Разработайте форму обратной связи с полями для ввода имени, email, сообщения.
   - Добавьте валидацию полей (например, проверку на корректность email) и отправку данных на сервер (можно использовать фейковый API).

5. Реализация маршрутизации в приложении:
   - Создайте несколько страниц в вашем приложении (например, домашняя страница, страница о нас, контакты).

- Используйте библиотеку React Router для организации навигации между этими страницами.

6. Аутентификация и авторизация:
   - Разработайте механизм аутентификации пользователей (например, форма входа).
   - Реализуйте защиту определенных страниц доступом только для авторизованных пользователей.

Дисциплина 4. «Основы backend-разработки на Node.js»

1. Создание сервера на Node.js. Напишите простой HTTP-сервер на Node.js, который возвращает "Hello, World!" при обращении к корневому URL.

2. Работа с маршрутами. Реализуйте несколько маршрутов (routes) в вашем приложении на Node.js для обработки GET и POST запросов.

3. Взаимодействие с базой данных. Создайте базу данных SQLite и напишите скрипт на Node.js для создания таблицы и выполнения простых CRUD-операций.

4. Использование фреймворка Express. Перепишите ваше приложение на Node.js, используя фреймворк Express для обработки маршрутов и запросов.

5. Асинхронное программирование. Напишите несколько примеров кода на Node.js, демонстрирующих использование колбеков, промисов и async/await для работы с асинхронными операциями.

Дисциплина 5. «Базы данных»

1. Создание простого API. Создать API с использованием Node.js и PostgreSQL для управления информацией о студентах. Задание включает в себя создание эндпоинтов для добавления, удаления, обновления и получения информации о студентах из базы данных.

2. Работа с транзакциями. Создать приложение, используя Node.js и PostgreSQL, которое демонстрирует использование транзакций для обеспечения целостности данных при выполнении нескольких операций.

3. Оптимизация запросов: Выполнить задание по оптимизации запросов к базе данных PostgreSQL с использованием индексов, оптимизированных запросов и других методов для улучшения производительности.

Дисциплина 6. «Фреймворк NestJS»

1. Создание простого REST API. Напишите приложение на NestJS, которое предоставляет REST API для управления списком пользователей. API должно поддерживать операции CRUD (создание, чтение, обновление, удаление).

2. Использование Middleware. Реализуйте middleware в вашем приложении NestJS для логирования запросов и ответов, а также для проверки аутентификации пользователей.

3. Использование Swagger для документирования API. Интегрируйте Swagger в ваше приложение NestJS для автоматической генерации документации API. Добавьте описания маршрутов, параметры запросов и примеры ответов.

4. Работа с базой данных. Подключите базу данных PostgreSQL или MongoDB к вашему приложению NestJS и реализуйте сервисы для взаимодействия с этой базой данных (например, сохранение пользователей или их профилей).

Дисциплина 7. «GNU/Linux, Nginx, Docker»

1. Установка и настройка Nginx на сервере. Установите Nginx на виртуальной машине с ОС GNU/Linux. Настройте веб-сервер Nginx для

обслуживания статических файлов и настройте виртуальные хосты для нескольких веб-сайтов.

2. Создание Docker-контейнера для приложения. Напишите Dockerfile для создания образа, который будет содержать простое веб-приложение. Затем создайте и запустите контейнер на основе образа, полученного из этого Dockerfile.

3. Настройка обратного прокси с использованием Nginx. Настройте Nginx в качестве обратного прокси для передачи запросов к backend-ам, работающим в контейнерах Docker, запущенных на том же сервере. Протестируйте обратный прокси для нескольких веб-приложений, запущенных в контейнерах.

4. Масштабирование приложения с использованием Docker Compose. - Используйте Docker Compose для определения и запуска множества контейнеров, включая веб-сервер Nginx и несколько экземпляров веб-приложения.

5. Настройка SSL и обеспечение безопасности с помощью Nginx. Настройте Nginx для использования SSL-сертификатов и настройте перенаправление с HTTP на HTTPS.

6. Деплой приложения. Задокументируйте процесс деплоя вашего backend-приложения на хостинге или в облаке, используя Docker или другие инструменты. При возможности, создайте bash-скрипт пакетного развёртывания приложения вашего приложения на “чистой” облачной операционной системе.

Критерии оценки текущих практических заданий по дисциплинам.

На основании выполненных практических заданий обучающемуся определяется оценка – «зачтено», «не зачтено».

Оценка	Уровень подготовки
Зачтено	Хорошая подготовка. Обучающийся выполнил все предложенные задания.
Не зачтено	Подготовка недостаточная и требует дополнительного изучения материала.

Пример описания итогового зачетного проекта

Работа в качестве Fullstack-разработчика на JavaScript, React и Node.js на стажировке в ООО «Академия Лад».

Критерии оценки итогового проекта

На основании выполненного итогового проекта обучающемуся определяется оценка – «зачтено», «не зачтено».

Оценка	Уровень подготовки
Зачтено	Хорошая подготовка. Обучающийся выполнил все предложенные этапы работы над проектом. Проведено исследование пользовательского опыта, представлены его результаты. Сделаны корректные выводы.
Не зачтено	Подготовка недостаточная и требует дополнительного изучения материала.

Литература

Дисциплины 1-3

- Книги

1. "HTML и CSS. Дизайн и построение веб-сайтов" Джон Даккетт - Перевод книги, которая поможет понять основы HTML и CSS.
2. "CSS. Технология создания стилей для профессионалов" Эрик Мейерс - Книга о продвинутых техниках работы с CSS.
3. "JavaScript. Шаблоны" Стоян Стефанов - Книга о шаблонах программирования на JavaScript.
4. "JavaScript. Подробное руководство" Дэвид Флэнаган - Классическое руководство по JavaScript.

5. "React. Быстрый старт" Пол Картер - Книга, которая поможет вам начать работу с React.

- **Источники в интернете**

1. MDN Web Docs (https://developer.mozilla.org/): Официальный ресурс от Mozilla, который предлагает подробные и понятные статьи по HTML, CSS и JavaScript.
2. CSS-Tricks (https://css-tricks.com/): Этот сайт предлагает множество полезных статей, уроков и ресурсов по CSS.
3. learn.javascript (https://learn.javascript.ru/): Интерактивный учебник по JavaScript.
4. React.js документация (https://ru.legacy.reactjs.org/): Здесь вы найдете подробные инструкции и примеры использования React.
5. W3Schools (https://www.w3schools.com): Этот ресурс предлагает простые примеры и уроки по HTML, CSS, JavaScript и другим технологиям веб-разработки.
6. Stack Overflow (https://stackoverflow.com/): Форум для программистов, где можно найти ответы на многие вопросы по HTML, CSS, JavaScript и React.

Дисциплины 4-7

- Книги

1. "Node.js в действии" - Энтони С. Дж. (Anthony S. J. Dahlin), Луис Атенсио (Luis Atencio)

2. "Высоконагруженные приложения: разработка, тестирование, масштабирование" - Мартин Клебаум (Martin Kleppmann)

3. "PostgreSQL: профессиональное программирование на SQL" - Томас Лок (Thomas Lock)

4. "Nest.js: создание масштабируемых и эффективных

веб-приложений" - Дэвид Джонсон (David J. Johnson)

5. "Linux для чайников" - Ричард Блум (Richard Blum)

6. "Nginx. HTTP-сервер высокой производительности" - Клемент Иванов (Clement Ivanov)

7. "Docker: Построение и развертывание приложений" - Пабло Кантизани (Pablo Kanziani)

8. "JavaScript. Подробное руководство" - Дэвид Флэнаган (David Flanagan)

9. "SQL и реляционные базы данных: учебное пособие" - Александр Болдырев

10. "Операционная система Linux" - Виктор Горшков.

- **Источники в интернете**

1. MDN Web Docs (https://developer.mozilla.org) - отличный ресурс для изучения JavaScript, Node.js и других веб-технологий.

2. Node.js документация (https://nodejs.org/en/docs/) - официальная документация Node.js, предоставляющая обширную информацию по различным аспектам использования Node.js.

3. PostgreSQL документация (https://www.postgresql.org/docs/) - официальная документация PostgreSQL, содержащая разделы по установке, использованию и настройке базы данных.

4. NestJS документация (https://docs.nestjs.com) - официальная документация фреймворка NestJS, предоставляющая руководства и примеры использования.

5. Linux.org (https://www.linux.org) - ресурс с обширной информацией о операционной системе Linux, включая руководства и форумы сообщества.

6. Nginx документация (https://nginx.org/en/docs/) - официальная документация Nginx, содержащая руководства по конфигурации и

использованию веб-сервера.

7. Docker документация (https://docs.docker.com) - официальная документация Docker, содержащая руководства по установке, использованию и развертыванию контейнеров.

8. W3Schools (https://www.w3schools.com) - ресурс с обширной информацией по веб-технологиям, включая HTML, CSS, JavaScript и другие.

9. Stack Overflow (https://stackoverflow.com) - популярный форум, где разработчики могут задавать вопросы и находить ответы на широкий спектр тем, связанных с программированием.

10. GitHub (https://github.com) - платформа для хранения кода, где можно найти множество открытых проектов, библиотек и примеров кода, связанных с backend-разработкой на Node.js и другими технологиями.

6. ОРГАНИЗАЦИОННО-ПЕДАГОГИЧЕСКИЕ УСЛОВИЯ ОБЕСПЕЧЕНИЯ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА

Руководитель программы профессиональной переподготовки:

Усков А.В. Директор ООО «Академия Лад»

Разработчики программы профессиональной переподготовки:

Усков А.В. Директор ООО «Академия Лад»

Слугин В.Г. Начальник вычислительного центра ГБПОУ «НРТК»

Составители учебно-тематического плана программы профессиональной переподготовки:

Усков А.В. Директор ООО «Академия Лад»

Куликов Д.И. Старший инженер программист ООО Студия Т_Г

Слугин В.Г. Начальник вычислительного центра ГБПОУ «НРТК»

Сведения о педагогических (научно-педагогических) работниках, участвующих в реализации программы профессиональной

переподготовки, и лицах, привлекаемых к реализации дополнительной образовательной программы на иных условиях, представлены в таблице 2.

Таблица 2

Преподаватели программы профессиональной переподготовки «Fullstack-разработка на JavaScript»

№ п/п	Наименование дисциплин (модулей)	Фамилия, имя, отчество,	Ученая степень, ученое звание	Основное место работы, должность	Место работы и должность по совместительству (если есть)
1.	Дисциплина «Вёрстка сайтов»	Усков А.В.		Директор ООО «Академия Лад»	
2.	Дисциплина «Программирование на JavaScript»	Лушкин С.Н.		TastyTeam, frontend-разработчик	
3.	Дисциплина «Frontend-разработка на React»	Куликов Д.И.		Старший инженер программист ООО Студия Т_Г	
4.	Дисциплина «Основы backend-разработки на Node.js»	Слугин В.Г.		Начальник вычислительного центра ГБПОУ «НРТК»	Зам. директора по образовательным проектам, программист ООО «ГК АЗЪ»
5.	Дисциплина «Базы данных»	Марков А.Н.		ГК Lad, backend-разработчик	
6.	Дисциплина «Фреймворк NestJS»	Слугин В.Г.		Начальник вычислительного центра ГБПОУ «НРТК»	Зам. директора по образовательным проектам, программист ООО «ГК АЗЪ»
7.	Дисциплина «GNU/Linux, Nginx, Docker»	Слугин В.Г.		Начальник вычислительного центра ГБПОУ «НРТК»	Зам. директора по образовательным проектам, программист ООО «ГК АЗЪ»

7. МАТЕРИАЛЬНО-ТЕХНИЧЕСКИЕ УСЛОВИЯ РЕАЛИЗАЦИИ ПРОГРАММЫ

№ п.п.	Наименование дисциплины (модуля)	Материально-технические условия для реализации программ (наличие лабораторий, производственных участков и т.п. по профилю программы профессиональной переподготовки)
1.	Дисциплина «Вёрстка сайтов»	- Система дистанционного обучения: https://getcourse.ru
2.	Дисциплина «Программирование на JavaScript»	
3	Дисциплина «Frontend-разработка на React»	
4.	Дисциплина «Основы backend-разработки на Node.js»	
5.	Дисциплина «Базы данных»	
6.	Дисциплина «Фреймворк NestJS»	
7.	Дисциплина «GNU/Linux, Nginx, Docker»	

УЧЕБНЫЙ ПЛАН

ПРОГРАММА ПРОФЕССИОНАЛЬНОЙ ПЕРЕПОДГОТОВКИ

«Fullstack-разработка на JavaScript»

№ пп	Наименование модулей	Всего, час.		В том числе		
			Аудиторных	Лекции	Семинары, практические занятия	Самостоятельная работа
1.	**Модуль 1. Верстка сайтов**					
	Основы html и css	11	6	2	4	5
	Codestyle и codeguide	3	2	1	1	1
	CSS flexbox	8	5	1	4	3
	CSS Grid	7	5	1	4	2
	БЭМ — методология	5	4	2	2	1
	Адаптация сайтов под мобильные устройства	6	5	1	4	1
	Автоматизация верстки	4	3	2	1	1
	Промежуточная аттестация	2				
	ИТОГО	**46**	**30**	**10**	**20**	**14**
2.	**Модуль 2. Программирование на JavaScript**					
	Введение в JavaScript. Переменные и типы данных	4	3	1	2	1
	Git. Настройка среды. Преобразование типов. Операторы сравнения	5	3	1	2	2
	Условные и логические операторы, циклы	5	3	1	2	2
	Функции	5	3	1	2	2
	Массивы и их методы, объекты	9	6	2	4	3
	Продвинутая работа с функциями, ключевое слово this, контекст, замыкания	5	3	1	2	2
	Прототипы и классы	5	3	1	2	2
	Document Object Model	5	3	1	2	2

	Обработка ошибок. Fetch	3	3	1	2	
	Промежуточная аттестация	2				
	ИТОГО	**48**	**30**	**10**	**20**	**16**
3.	**Модуль 3. Frontend-разработка на React**					
	Инициализация react-приложения, структура приложения и работа с пакетным менеджером npm. Настройка рабочей среды.	4	3	1	2	1
	Базовые концепции React	4	3	1	2	1
	Функциональные компоненты.	4	3	1	2	1
	Знакомство с понятиями props и state	4	3	1	2	1
	Основные этапы и методы жизненного цикла компонентов.	4	3	1	2	1
	Классовые и функциональные компоненты.	4	3	1	2	1
	Знакомство с React-Hooks	4	3	1	2	1
	Роутинг в react-приложении	4	3	1	2	1
	Подключение и настройка менеджера состояний Redux	4	3	1	2	1
	ReactDevTools и ReduxDevTools	4	3	1	2	1
	http-клиент axios.	3	3	1	2	
	Рендер React приложений	3	3	1	2	
	Промежуточная аттестация	2				
	ИТОГО	**48**	**36**	**12**	**24**	**10**
4.	**Модуль 4. Основы backend-разработки на Node.js**					
	Введение в курс. Язык JavaScript. Рантаймы исполнения языка: браузер, платформа Node.js. Установка инструментария для старта. Первый проект. Репозиторий GIT.	3	3	1	2	0
	Разделение кода Node.js пакета на модули. Разновидности модулей: CommonJS modules, ECMAScript modules	3	3	1	2	0
	Ввод/вывод данных в	4	3	1	2	1

	Node.js-приложениях: STDIN, STDOUT, STDERROR.					
	Работа с переменными окружения. IPC (межпроцессное взаимодействие)	4	3	1	2	1
	Протокол HTTP(S). Создание простого веб-сервера на Node.js	4	3	1	2	1
	Создание CRUD API с имитацией базы данных. Основы работы с инструментами Postman и curl	4	3	1	2	1
	Создание backend приложения на фреймворке Express. Интеграция в приложение примера frontend кода.	4	3	1	2	1
	Основы работы с сервером VPS на Ubuntu 22.04 для публикации приложения на Node.js	4	3	1	2	1
	Знакомство с Docker и Docker-Compose.	4	3	1	2	1
	Промежуточная аттестация	2				
	ИТОГО	**36**	**27**	**9**	**18**	**7**
5.	**Модуль 5. Базы данных**					
	Основы работы с ORM Sequelize.	5	3	1	2	2
	Подключение базы данных к backend приложению на фреймворке Express посредством ORM Sequelize.	5	3	1	2	2
	Основы реляционных баз данных. Нормализация баз данных. Основы языка SQL	6	4	2	2	2
	Проектирование реляционной базы данных.	6	4	2	2	2
	Проектирование backend приложения на фреймворке Express. Аутентификация JWT. Добавление аутентификации в backend и frontend.	6	4	2	2	2
	Основы TypeScript. Инициализация и сборка проекта.	6	4	2	2	2
	Промежуточная аттестация	2				
	ИТОГО	**36**	**22**	**10**	**12**	**12**
6.	**Модуль 6. Фреймворк NestJS**					

	Введение в NestJS. CLI-инструментарий для автоматизации управления проектом.	3	3	1	2	0
	Добавление объектно-реляционного отображения (ORM) TypeORM к приложению на NestJS.	3	3	1	2	0
	Создание проекта backend приложения на NestJS. Хеширование пароля пользователя. API для сопоставления ролей пользователям.	4	3	1	2	1
	Добавление аутентификации JWT в NestJS приложение.	4	3	1	2	1
	Обеспечение безопасности VPS сервера на Ubuntu 22.04 для публикации приложения на NodeJS (ssh, ssh-keygen, ssh config, port forwarding, firewall, docker).	4	3	1	2	1
	Реверс инжениринг существующей базы данных посредством модуля typeorm-model-generator. Работа с сущностями (Entityes) TypeORM через "Repository"	4	3	1	2	1
	Работа с миграциями TypeORM в приложении на NestJS.	4	3	1	2	1
	Интеграция NestJS с OpenAPI(Swagger)	4	3	1	2	1
	Кратко о front-end фреймворке Angular: Схожесть архитектуры исходного кода приложений с backend фреймворком NestJS. Схожесть CLI-инструментария. Пример создания простого SPA, взаимодействующего с сервером.	4	3	1	2	1
	Промежуточная аттестация	2				
	ИТОГО	**36**	**27**	**9**	**18**	**7**
7.	**Модуль 7. GNU/Linux, Nginx, Docker**					
	Сборка дистрибутива backend и frontend приложения. Особенности публикации fullstack приложений в GNU/Linux. Systemd service.	5	3	1	2	2
	Настройка Postgresql на Ubuntu	4	3	1	2	1

	GNU/Linux. Конфигурирование systemd сервиса для запуска full-stack приложения с учётом миграций.					
	Основый языка сценариев bash. Создание сценария (скрипта) автоматического развёртывания Fullstack приложения на Nestjs+Postgres на Ubuntu GNU/Linux.	4	3	1	2	1
	Установка и настройка веб-сервера Nginx в операционной системе GNU/Linux. Использование Nginx как сервер для статического контента frontend. Использование Nginx как Reverse-Proxy сервер для back-end приложения на Nodejs.	4	3	1	2	1
	Виртуальные хосты в NGINX. NGINX в режиме SSL. Openssl. 301 редирект с HTTP на HTTPS. Lestencrypt. Certbot	4	3	1	2	1
	Docker. Образы и контейнеры. Основные команды управления. Публикация образов на dockerhub.	4	3	1	2	1
	Docker-Compose. Создание набора контейнеров для функционирования fullstack приложения, работающего с базой данных.	4	3	1	2	1
	Работа с GitHub Actions для сборки и доставки приложений на сервер (элемент CI/CD)	5	3	1	2	2
	ИТОГО	**34**	**24**	**8**	**16**	**10**
	Итоговая аттестация	**4**				
	ИТОГО ПО ПРОГРАММЕ	**288**	**196**	**68**	**128**	**80**

Календарный учебный график

Нед	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52
I	Т	Т	Т	Т	Т	Т	Т	Т	Т	Т	Т	Т	Т	Т	Т	Т	Т	Т	Т	Т	Т	Т	Т	Т	Т	Т	Т	Т	Т	Т	Т	Т	Т	Т	Т	Т	Т	Т	Т	Иа												
	ПС	ПС	ПС	ПС	Па	ПС	ПС	ПС	ПС	Па	ПС	ПС	ПС	ПС	Па	ПС	ПС	ПС	ПС	Па	ПС	ПС	ПС	ПС	Па	ПС	ПС	ПС	ПС	Па	ПС	ПС	ПС	ПС	Па	ПС	ПС	ПС	ПС	*												
	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*												
	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*												
	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*												
	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*												
	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*												

Т	Теоретическое обучение
ПС	Практические семинарские занятия
Па	Промежуточная аттестация
Иа	Итоговая аттестация
К	Каникулы
*	Обучение не ведется