Публикация на тему

# Разработка проектов в обучающем процессе переподготовки студентов на базе высшего образования

*Для студентов и преподавателей высших учебных заведений направлений связанных с разработкой проектов, программирование, тестированием и дизайном.*

**Анотация**

В публикации подробно рассматривается этапы разработки проекта, начиная от проведения социологических опросов и дизайна до поддержки и тестирования готового приложения.

The publication details the stages of project development, from conducting opinion polls and design to supporting and testing the finished application.

**Автор**

Михалькевич Александр Викторович

## Публикация

**Наименование** Разработка проектов в обучающем процессе переподготовки студентов на базе высшего образования

**Автор** А.В.Михалькевич

**Специальность** Для студентов и преподавателей высших учебных заведений направлений связанных с разработкой проектов, программирование, тестированием и дизайном.,

**Анотация**

В публикации подробно рассматривается этапы разработки проекта, начиная от проведения социологических опросов и дизайна до поддержки и тестирования готового приложения.

**Anotation in English**

The publication details the stages of project development, from conducting opinion polls and design to supporting and testing the finished application.

**Ключевые слова** разработка web-приложения, разработка веб-приложения, разработка проектов, образование, дисциплины, высшие учебные заведения, тестирование, дизайн, программирование

**Количество символов** 32604

# Содержание



# Введение

# 1 Этапы разработки проекта на примере web-приложения

Для примера возьмём стандартный проект - web-приложение.

Важно понимать, что разработка web-приложения начинается не тогда, когда программист начинает писать код, а раньше, с постановки целей и задач. После определения целей и задач проекта, необходимо выявить фокусную группу потенциальных пользователей и провести среди них социологический опрос. Затем подготовить базу данных. Затем, на основании полученных данных разработать макет приложения. Потом начать с разработки бэкенд части. Когда бэкенд будет формировать данные для главной страницы и другую основу для фронтенда (например, авторизацию), тогда можно переключаться на фронт. С этого момента начинается тесное взаимодействие бэка и фронта. Суть разработки web-приложения в этом и заключается - паралельная разработка фронтенд и бэкенд частей. Бэк и фронт должны уметь тесно взаимодействовать друг с другом. На стороне фронта формируется визуализация и отправка запросов на бэк. А на бэке - формируется ответ фронту (чаще всего и удобнее в формате JSON, который понимают все современные фронтовые фрэймворки). Т.е. Request - это то, что должен отправить фронт, и что принимает бэк. Response - это то, что отправляет бэк, и на основании чего фронт формирует вид. Часто это взаимодействие бывает сложным, и на этом этапе необходимо внедрять тестирование и дисциплины управления проектом.

Web-приложение должно быть размещено на сервере, который тоже необходимо обслуживать. И особенностью разработки web-приложений (в сравнении со стационарными приложениями) является то, что web-приложение должно разрабатываться и поддерживаться постоянно. Следующие этапы разработки: разрабтка бэка, разработка фронта, тестирование,

обслуживание сервера и управление проектом - это постоянный процесс.

Итак, выделим основные этапы разработки web-приложения.

- формирование команды
- определение целей и задач
- выявление фокусной группы пользователей
- проведение социологических опросов среди выявленной группы
- формирование дизайна на основе социологических опросов
- вёрстка макета главной страницы приложеня
- подготовка сервера
- создание базы данных
- бэкенд (подключение и последующая разработка)
- фроненд (взаимодействие с бэкендом и последующая разработка)
- тестирование
- управление проектом.

# 1 .1 Начало разработки

Сперва формируется команда, внутри которой идёт обсуждение проекта. При этом, каждый участник проекта должен представлять общую конечную цель разработки, а также сферу своих задач и задач других участников команды, иметь возможность обсуждать задачи с другими участниками.

**Формирование команды**

В команду разработчиков web-приложения входят следующие специалисты:

- дизайнер (менеджер, product-owner, team-lead, или другой ответственный за существование проекта и за макеты человек);
- бэкенд разработчик;
- фронтенд разработчик;
- тестировщик.

Кроме того, необходим специалист по обслуживанию серера приложения, devops. Но, т.к. один сервер может обсуживать множество web-приложений (соответственно и команд), то наличие девопса в команде не обязательно. Конечно, при условии, что у команды есть доступ к настройкам сервера.

Команда разработчиков может состоять из: 1, 4, 5, 8 человек.

1. Если команда состоит из одного человека - то это должен быть web-мастер, на которого возлагаются задачи по фронтенду, бэкенду, дизайну и тестированию.

4. Если команда состоит из 4 человек - то роли распределяются следующим образом: фронтенд-разработчик, бэкенд-разработчик, тестировщик и дизайнер.

5. Если команда состоит из 5 человек - то добавляется еще роль руководителя (это может быть управляющий проектом, маркетолог, тимлид, менеджер или контент-менеджер). Это человек, который руководит проектом и направляет разработку.

8. Если команда состоит из 8 человек - то это два фронтенд-разработчика, два бэкенд-разработчика, два дизайнера и два тестировщика. Роль управляющего проектом возлагается на

одного из члена команды.

Важно отметить, что команды из двух или трёх разработчиков не бывает. В противном случае, одному или нескольким специалистам пришлось бы в совмещать противоположные по смысловой нагрузке роли. Например, специалист может знать фронтенд и уметь рисовать дизайн. Но если в команде он будет выполнять роль дизайнера и фронтендера, то он будет подстраивать дизайны под себя, а не под пользователя (подсознательно или осознанно). Тоже и с тестированием: можно быть или разработчиком, или тестировщиком, но не совмещать эти роли. Если разработчик одновременно является и тестировщиком, то у команды нет тестировщика.

**От идеи до макета**

Начало разработки web-приложения - это период времени с оформления идеи (во время контакта с заказчиком или в кругу доверенных лиц) до появления готового макета главной страницы.

Этот этап должен включать в себя проведение социологических опросов и их учитывание в макете, чтобы не получилась разработка под одного человека. Если дизайнеры конструируют макет в соответствии со своими предположениями, не посчитав нужным их проверить, то может оказаться, что их предположения могут быть не верны. Вообще отсутствие социологических опросов при моделировании дизайна, может привести к роковым ошибкам, которые не дадут проекту появиться:

- отстутствие в дизайне концептуальной модели

- отсутствие целевой аудитории (или несоответствие выбранного дизайна целевой аудитории)

Итак, если команда, целевая аудитория и концептуальная модель определены, то можем переходить к следующим этапам разработки.

## 1 .2 Этапы разработки

Итак, после того, как идея web-приложения обсуждена, всем участником команды понятен конечный результат и соблюдены прочие условия, необходимые для старта проекта, можно приступать к следущим этапам разработки.

Этапы разработки web-приложения:

Заведение аккаунтов в следующих системах: Google, Discord, Github, Erud.by, Figma, Amazon.

Установка программного обеспечения: web-браузера с набором пакетов для разработки, интегрированная среда разработки PHPStorm, система контроля версий git с пользовательским интерфейсом SublimeMerge, локального сервера для языка программирования PHP, системы развёртывания интерфейсов Docker, инструмент для тестирования Postmen.

Разработка макета главной страницы для будущего web-приложения. Разработка макета ведётся в программе Figma. При необходимости разрабатывается несколько адаптивных вариантов макета главной страницы или другие страницы web-приложения.

Вёрстка макета с использованием HTML и CSS. Изучение библиотек Bootstrap и jQuery. Использование готовых шаблонов Bootstrap.

Подключение базы данных MySQL.

Разработка серверной части, бэкенд API для web-приложения. Для разработки серверной части приложения воспользуемся языком программирования PHP и фрэймворком Laravel. Задачи возгалаемые на серверный фрэймворк: преоборазование данных из MySQL в JSON, авторизация и аутенификация пользователей в системе, система администрирования, сохранение пользовательских данных, отправка сообщений пользователю, прослушка пользовательских событий, группировка задач в программные очереди, серверная маршрутизация и другие серверные задачи.

Разработка клиентской части web-приложения. С использованием фронтенд фрэймворка Vue. Задачи возлагаемые на фронтенд фрэймворк: клиентская маршрутизация, формирование интерфейса взаимодействия с пользователем, шаблонизация web-приложения, разрабтка компонентов, вывод данных из формата JSON по серверным маршрутам, шаблонизация проекта и разбивка шаблонов на многократно-используемые компоненты. Для решения поставленных задач используется JavaScript и Node.js, пакетный менеджер NPM, препроцессоры SASS, SCSS.

Тестирование. С использованием инструмента Postmen.

Использование систем управления проектом и систем постановки задач. Jira, Bitrix24.

Внедрение новых идей в проект. При этом далее паралельно ведётся разрработка бэкенд и фронтенд частей, их взаимодействие и тестирование. Так, в отличии от десктопных программ, разработка web-приложения — это бесконечный процесс.

# 2 Процесс разработки

Процесс разработки web-приложения подразумевает непрерывное взаимодействие бэкенд, фронтенд и тестирования.

Первоначальная задача, которая возлагается на фронтенд - это вёрстка главной страницы. И интегрирование вёрстки в фронтенд фрэймворк. Будем использовать фрэймворк **Vue3**.

Первоначальная задача, возлагаемая на бэкенд - это создание базы данных. И интегрирование базы с бэкенд фрэймворком. Будем использовать базу данных **MySQL** и фрэймворк **Laravel**.

В кратце, дальнейший процесс разработки web-приложения сводится к следующему. Постановка задачи, создание дизайна, вёрстка, разработка фронтенд, разработка бэкенд, интеграция бэкенд и фронтенд, деплой в тестовую среду, тестирование, устранение ошибок, выкатывание на прод. И так далее, с другими задачами.

# 3 Инструментарий

Условно, используемый инструментарий можно разделить на два типа:

- общекомандный инструментарий
- индивидуальный

**Общекомандный** - это общий для команды разработчиков, тестировщиков инструментарий. Это система контроля версий, серверные технологии, база данных, платформа автоматизации развёртывания и управления приложением, инструменты тестирования, мессенджеры...

Рассмотрим примерный набор общекомандных инструментов (в зависимости от проекта и команды, инструменты могут отличаться от представленных ниже).

Система контроля версий - **git**, а также репозитории: github - для open source проектов и bitbacket, gitlab - для закрытых проектов.

Серерные технологии: **node**, **php**, а также сопутствующие инструменты, например для PHP - это сервер **Apache** и менеджер зависимостей **Composer**.

База данных: **MySQL**.

Платформа автоматизации развёртывания и управления приложением - **docker**.

Инструменты тестирования - **postmen**.

Мессенджеры - **discord**, **telegram**.

**Индивидуальный** - это операционная система, интегрированная среда разработки, текстовый редактор, браузер...

Интегрированная среда разработки - **PHPStorm** (для бэкенд разработки) и **WEBStorm** (для фронтенд разработки).

Текстовый редактор - любой на выбор.

Браузеры - **Firefox**, **Chrome**.

# 4 Бэкенд и фронтенд. Разработа и взаимодействие

Для разработки фронтенди и бэкенд частей web-приложения используются разные фрэймворки. На данный момент, наилучшие решение - это использование Laravel для бэкенд и Vue для фронтенд.

# 4 .1 Разработка бэкенд API с помощью фрэймворка Laravel

Разработка серверной части web-приложения ведётся с помощью архитектурного шаблона проектирования HMVC и объектно-ориентированных принципов программирования. И сводится к следующим этапам:

Установка фрэймворка Laravel.

Подключение к базе данных.

Выполнение первоначальных миграций.

Создание и выполнение миграций для статических страниц web-приложения.

Создание и выполнение сидов, первоначальных данных для web-приложения.

Создание моделей данных, использование моделей и для преобразования данных в JSON, использование конструктора запросов Eloquent.

Маршрутизация для статических страниц. Виды маршрутизаторов: web, api, console и

broadcast.

Бэкенд-авторизация с помощью модуля Sunctum.

Использование модуля Laravel/Socialite для авторизации через социальные системы.

Разработка динамических страниц и маршрутов для фронтенд.

Разработка CRUD-контроллеров приложения.

Разработка системы администриования.

## 4 .2 Разработка frontend API с помощью фрэймворка Vue

Разработка части фронтенд приложения ведётся с помощью шаблона проектирования MV-VM и функциональньного компонентного программирования.

Установка Node.js и менеджера пакетов NPM. Основы использования.

Установка Vue и Vue-cli.

Способы создания приложения: с помощью менеджера проектов и с помощью консольных команд Vue-cli.

Подключение свёрстанного шаблона к приложению Vue.

Шаблонизация проекта. Выделение базового шаблона и подшаблонов страниц. Динамические и статические страницы.

Разбивка шаблонов на компоненты с расширением .vue.

Подключение модулей Axios, Eslint, Babel и router-vue.

Фронтенд маршутизация.

Получение данных из бэкенд маршрутов с помощью модуля Axios.

Создание функций и прослушивателей на стороне фронтенд, разработка форм приложения.

Использование типов запросов Get, Post, Put, Delete и Any.

Использование модуля Vue\Socialite и модульная обработка бэкенд запросов.

Использование токена в запросах.

Сохранение данных на строне фронтенд.

Разработка кабинета пользователя.

# 5 Взаимосвязь дисциплин в двухгодичном курсе по специальности "Разработка web-приложений"

Двухгодичный курс обучения можно разбить на 4 части, или 4 семестра обучения. Каждый семестр должен соответствовать этапу в разработке web-приложения. Цель всего курса одна - это разработка качественного web-приложения, но на каждый этап возлагаются свои задачи и цели. Сперва рассмотрим цели и задачи каждого этапа:

1. **Начало разработки проекта**. Так на первом этапе студенты объединяются в команды, определяются с темой проекта и готовят дизайн своего будущего проекта.

2. **Бэкенд**. Задача второго этапа - реализация серверной части web-приложения: бэкенд и

база данных.

3. **Фронтенд**. Это этап вёрстки, клиентского программирования и тестированя web-приложения.

4. **Управление проектом**. Разработка web-приложения - это бесконечный процесс. И на этом этапе (когда web-приложение уже есть), необходимы дисциплины управления, как проектом, так и персоналом.

Все дисциплины курса связаны между собой.

Дисциплины первого семестра - "Дизайн", "Графика", "Юзабилити", "Основы визуального дизайна". Кроме того, на преподавателей возлагается задача по формированию команд.

Дисциплины второго семестра - "Базы данных", "Серверные технологии разработки" и "Тестирование". Серверная часть приложения реализуется с помощью языка программирования PHP. Студенты должны научиться создавать базу данных, таблицы, вставлять данные, удалять, редактировать и выводить по условиям (CRUD) и пользоваться агрегатными функциями SQL. На преподавателя дисциплины "Серверные технологии разработки" возлагается обязанность - научить студентов преобразовывать данные из базы данных в формат JSON, никаких других ответов (в том числе и в формате HTML) от сервера быть не должно. Команда разрабатывает RESTfull API, которое будет использовано фронтенд-фрэймворком на следующем этапе. Для разработки RESTfull API лучше использовать фрэймворк Laravel. Соответственно, кроме изучения основ PHP, необходимо изучение этого фрэймворка. На этом этапе пока еще нет полноценного web-приложения, но уже необходимо внедрять дисциплину "Тестирования" для тестирования запросов (request) и ответов сервера (response). Для тестирования запросов можно воспользоваться инструментом Postmen.

Дисциплины третьего семестра - "Вёрстка", "Проектирование динамических страниц", "Безопасность web-приложений" и "Программирование на языке Python". Данный этап начинается с вёрстки главной страницы, студенты изучают html, css и основы JavaScript. На дисциплине "Проектирование динамических страниц" студенты используют разработанный на предыдущем этапе API, изучают продвинутое использование JavaScript, Node.js и фронтенд-фрэймворк, например Vue. Язык программирования Python может быть использован для парсинга данных и для написания скриптов по взлому "Безопасность web-приложения".

Дисциплины четвёртого семестра - "Разработка web-приложений", "Управление проектом" и "Бизнес-анализ". Дисциплина "Разработка web-приложений" подразумевает более тесное взаимодейтсвие фронтенд и бэкенд частей приложения. Подразумевается углубленное изучение Laravel и Vue, а также взаимодействие этих фрэймворков. На дисциплине "Управление проектом" студенты изучают программные инструменты для управления проектом, например CRM Bitrix24 или Jira, а также и методологии гибкой разработки - Ajile и Scrum.

Другие дисциплины, от которых не зависит разработка проекта, могут быть добавлены в учебный процесс в любой семестр.

По окнчании курса команда должна презентовать и защитить свой проект. Каждый студент "защищает" свою часть разработки, в соответствии с назначенной ролью, но презентация может быть обще-командная.

# 6 Рекомендуемые темы к изучению

Рассмотрим рекомендуемые темы изучения по дисциплинам.

**Дисциплины дизайна**

Композиция

Цветоведение

Юзабилити

Figma

**Управление проектом**

Jira

Системы контроля версий

Git, github.com

Авторизация и работа в системах Google, Amazon и Битрикс-24

**Серверные технологии**

Основы серверов

Службы создания серверов и прослушивания портов

Облачные технологии: файловые хранилища и хостинги

Node.js

PHP

База данных MySQL

Фрэймворк Laravel и разработка API для бэкенд.

**Проектирование динамических страниц и фрэймворк Vue**

Основы Node.js и npm

Установка Vue, Vue-cli

Создание проекта с помощью менеджера и через консоль

Шаблонизация

Маршрутизация. Основные и дочерние роуты.

Создание компонентов: компоненты страниц и вспомогательные.

Формат данных JSON. Взаимодействие с данными бэкенда.

CRUD — выполнение запросов на создание, редактирование, вывод и удаление записей.

Формы. Работа с данными форм.

Продивнутая работа с компонентами. Компоненты с параметрами

Классы и интерфейсы.

Управление состоянием

Авторизация и аутентификация пользователей.

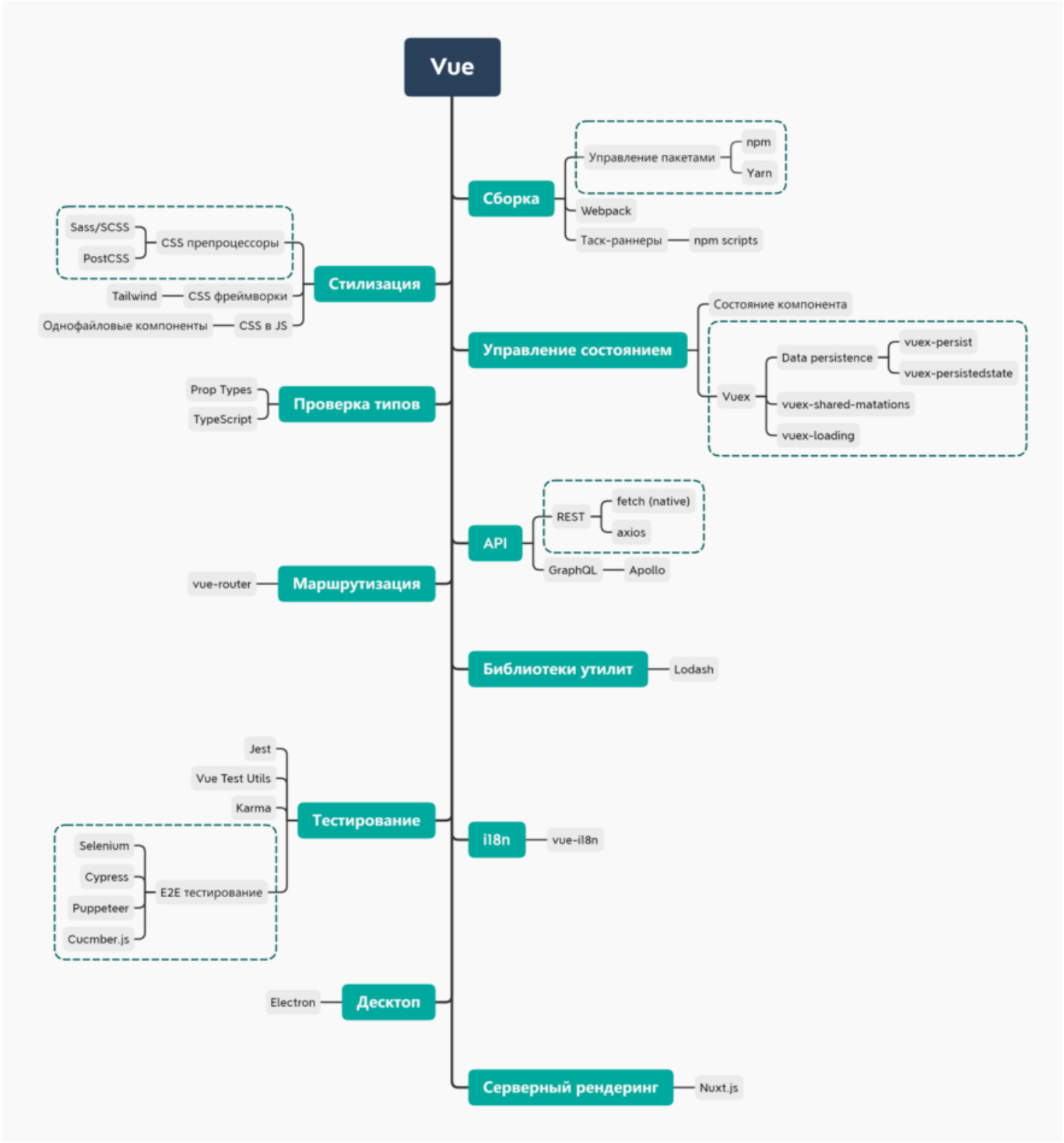
Vuetify.js

TypeScript

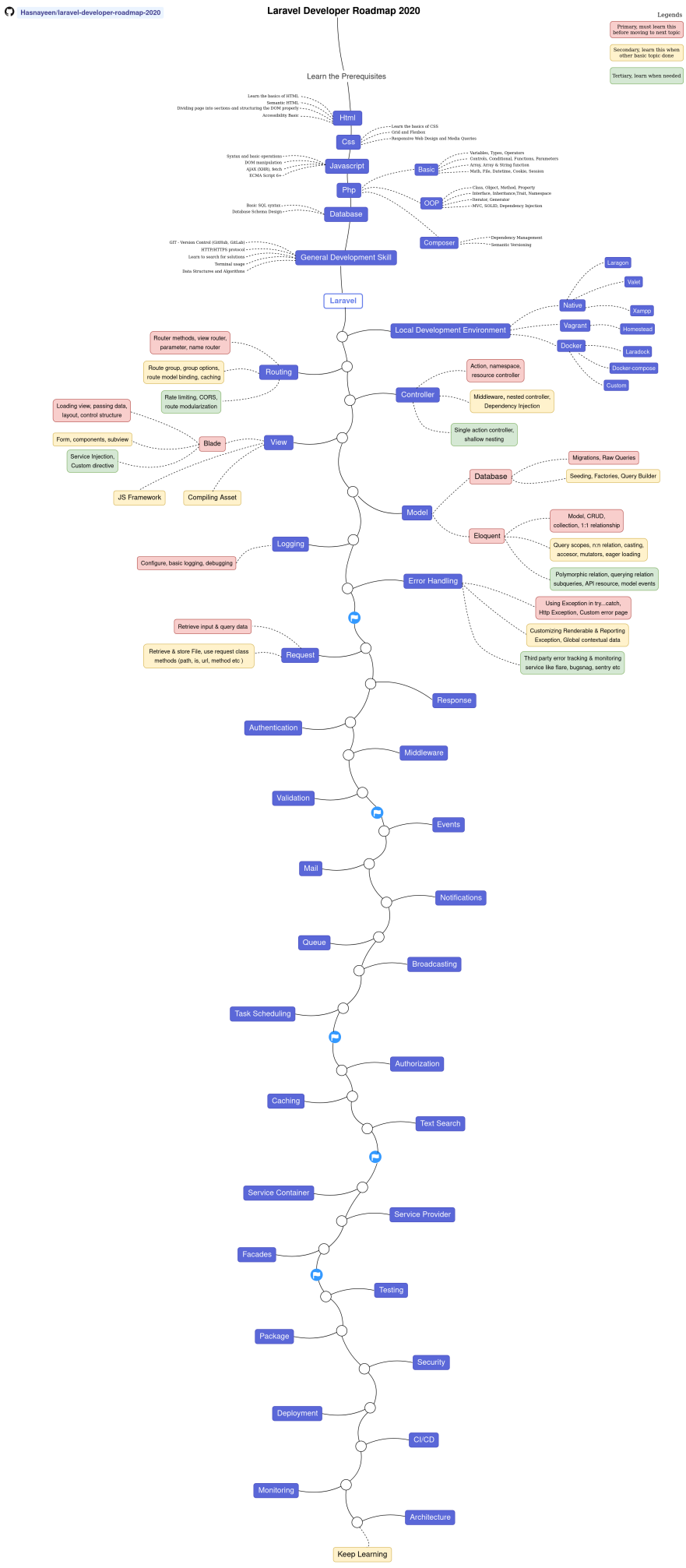
**Карты изучения**

Для некоторых технологий уже разработаны карты изучения.

Карта разработчика Vue:




Карта разработчика Laravel:


Hasnayeen/laravel-developer-roadmap-2020
Laravel Developer Roadmap 2020
Legends
Primary, must learn this before moving to next topic
Secondary, learn this when other basic topic done
Tertiary, learn when needed
Learn the Prerequisites
Learn the basics of HTML
Semantic HTML
Dividing page into sections and structuring the DOM properly
Accessibility Basic
Html
Learn the basics of CSS
Grid and Flexbox
Responsive Web Design and Media Queries
Css
Syntax and basic operations
DOM manipulation
AJAX (XHR), fetch
ECMA Script 6+
Javascript
Basic
Variables, Types, Operators
Controls, Conditional, Functions, Parameters
Array, Array & String Function
Math, File, Datetime, Cookie, Session
Php
OOP
Class, Object, Method, Property
Interface, Inheritance,Trait, Namespace
Iterator, Generator
MVC, SOLID, Dependency Injection
Composer
Dependency Management
Semantic Versioning
Basic SQL syntax
Database Schema Design
Database
GIT - Version Control (Github, GitLab)
HTTP/HTTPS protocol
Learn to search for solutions
Terminal usage
Data Structures and Algorithms
General Development Skill
Laravel
Local Development Environment
Laragon
Valet
Native
Xampp
Vagrant
Homestead
Docker
Laradock
Docker-compose
Custom
Router methods, view router, parameter, name router
Route group, group options, route model binding, caching
Rate limiting, CORS, route modularization
Routing
Controller
Action, namespace, resource controller
Middleware, nested controller, Dependency Injection
Single action controller, shallow nesting
Loading view, passing data, layout, control structure
Form, components, subview
Service Injection, Custom directive
Blade
View
JS Framework
Compiling Asset
Database
Migrations, Raw Queries
Seeding, Factories, Query Builder
Model
Eloquent
Model, CRUD, collection, 1:1 relationship
Query scopes, n:n relation, casting, accessor, mutators, eager loading
Polymorphic relation, querying relation subqueries, API resource, model events
Logging
Configure, basic logging, debugging
Error Handling
Using Exception in try...catch, Http Exception, Custom error page
Customizing Renderable & Reporting Exception, Global contextual data
Third party error tracking & monitoring service like flare, bugsnag, sentry etc
Retrieve input & query data
Retrieve & store File, use request class methods (path, is, url, method etc )
Request
Response
Authentication
Middleware
Validation
Events
Mail
Notifications
Queue
Broadcasting
Task Scheduling
Authorization
Caching
Text Search
Service Container
Service Provider
Facades
Testing
Package
Security
Deployment
CI/CD
Monitoring
Architecture
Keep Learning

## Заключение

## Список использованных источников

## Приложения