Титульный лист

МИНИСТЕРСТВО ОБРАЗОВАНИЯ МОСКОВСКОЙ ОБЛАСТИ

Государственное бюджетное профессиональное образовательное учреждение Московской области «Ногинский колледж»

Курсовой проект

по МДК.09.01 Проектирование и разработка веб-приложений

ПМ.09 Проектирование, разработка и оптимизация веб-приложений

Тема:

ИСПОЛЬЗОВАНИЕ ФРЕЙМВОРКА LARAVEL В РАЗРАБОТКЕ ВЕБ-ПРИЛОЖЕНИЯ «ФИНАНСОВАЯ КОНСУЛЬТАЦИЯ И КРЕДИТОВАНИЕ МАЛОГО БИЗНЕСА»

Разработчик:

студент группы 3ИСПР1

Нуштайкин Ю.Я.

(подпись)

Оценка защиты курсового проекта

«\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ »

Дата защиты

«\_\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024

Руководитель проекта:

преподаватель

Степанов С.О.

(подпись)

Ногинск, 2024 г

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc28135)

[Цель и задачи курсового проектирования 3](#_Toc30124)

[ОСНОВНАЯ ЧАСТЬ 5](#_Toc9235)

[1. Общие сведения о разработке веб-приложения на фреймворках Laravel и Bootstrap 5](#_Toc24871)

[2. Инсталляция всех необходимых программ и фреймворка, предпроектная подготовка к работе 9](#_Toc17601)

[3. Руководство программиста 13](#_Toc11876)

[ЗАКЛЮЧЕНИЕ 44](#_Toc17748)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 45](#_Toc4460)

[ПРИЛОЖЕНИЕ 2 46](#_Toc5405)

[ПРИЛОЖЕНИЕ 3 48](#_Toc16906)

# ВВЕДЕНИЕ

### **Цель и задачи курсового проектирования**

Целью данного курсового проекта была разработка веб-приложения на тему «Финансовой констультации и кредитования малого бизнеса». Основная задача - разработать и сверстать сайт, с работающим функционалом регистрации, авторизации, формой заявки для получения кредита. В условиях современного рынка малый бизнес играет ключевую роль в развитии экономики, создавая рабочие места и стимулируя инновации. Однако, зачастую предприниматели сталкиваются с проблемами, связанными с управлением финансами и доступом к кредитным ресурсам. Проект «Новые Бизнес Системы» предназначен для предоставления комплексных финансовых консультаций и облегчения процесса получения кредитов для малых и средних предприятий , личный кредит, а так же ипотека. NBS включает в себя следующие ключевые функциональные модули:

1. **Регистрация и авторизация** — безопасная система для создания учетной записи и входа пользователей на сайт, обеспечивающая конфиденциальность данных. А так же был добавлен функционал подтверждения аккаунта с помощью Email, смена почты, никнейма, пароля.

**2)Форма заявки на кредит** — удобная и интуитивно понятная форма, позволяющая предпринимателям быстро и легко подать заявку на получение кредита, ипотечного или личного займа.

**3) Кредитный калькулятор** — инструмент, помогающий пользователям рассчитать оптимальные условия кредитования, исходя из их финансовых возможностей и потребностей.

**4)** Роли:пользователь и администратор

Пользователь: любой человек прошедший регистрацию/авторизацию. Может отправить заявку на кредит, а так же имеет полный доступ к функционалу калькулятора, и всем фишкам связанным с его профилем.

Администратор: человек имеющий весь выше перечисленный функционал, но к нему добавляется панель администратора, в которой он имеет доступ к просмотру всех заявок на кредит.

# ОСНОВНАЯ ЧАСТЬ

### 1. Общие сведения о разработке веб-приложения на фреймворках Laravel и Bootstrap

Для разработки веб-приложения «NBS» используются фреймворки Laravel и Bootstrap. Они предоставляют мощные инструменты для создания современных и функциональных веб-приложений.

Laravel — это современный PHP-фреймворк, созданный для упрощения разработки веб-приложений. Он предоставляет широкий спектр инструментов и возможностей, которые делают процесс создания приложений более быстрым и эффективным. Как отмечается на официальном сайте Laravel: "Laravel стремится к созданию приложений, которые используют элегантный и выразительный синтаксис." Основные задачи и особенности Laravel включают:

**1) Богатый функционал:** Laravel предоставляет множество встроенных функций, таких как аутентификация, маршрутизация, управление сессиями, обработка запросов и ошибок, что делает его мощным инструментом для создания веб-приложений. Эти функции позволяют разработчикам сосредоточиться на логике приложения, не тратя время на рутинные задачи.

**2) Архитектура MVC:** Laravel использует архитектуру "Model-View-Controller" (MVC), что способствует разделению логики приложения и его представления. Это делает код более организованным, читабельным и легким в сопровождении. MVC позволяет легко масштабировать приложение, добавляя новые функции без нарушения существующей логики.

**3) Миграции и управление базой данных:** Laravel предлагает удобные инструменты для работы с базами данных, включая миграции, схемы и сиды. Миграции позволяют легко управлять версионностью базы данных, обеспечивая плавные обновления и откаты изменений. ORM (Eloquent) делает работу с базой данных интуитивно понятной и эффективной.

**4) Шаблонизация с Blade:** Laravel использует собственный движок шаблонов Blade, который позволяет создавать динамические интерфейсы с минимальными усилиями. Blade обеспечивает возможность использования PHP-кода внутри шаблонов, а также включает множество полезных директив, упрощающих работу с шаблонами.

**5) Интеграция с другими инструментами:** Laravel легко интегрируется с различными библиотеками и инструментами, такими как PHPUnit для тестирования, Redis для кэширования и очередей, и многими другими. Это делает Laravel гибким и расширяемым, позволяя разработчикам использовать лучшие инструменты для решения конкретных задач.

**6) Сообщество и экосистема:** Laravel обладает большим и активным сообществом, которое постоянно вносит свой вклад в развитие фреймворка. Существует множество готовых решений, пакетов и расширений, которые можно использовать в проектах. Официальный репозиторий пакетов Laravel (Packagist) предоставляет доступ к тысячам полезных библиотек.

Bootstrap — это популярный фронтенд-фреймворк, разработанный для создания адаптивных и удобных в использовании интерфейсов. Он предоставляет обширный набор предопределенных стилей и компонентов, что позволяет создавать привлекательные и функциональные решения с минимальными усилиями.

**Готовые компоненты:** Bootstrap предоставляет множество готовых компонентов, таких как кнопки, формы, навигационные панели, карточки, шаблоны и примеры, которые можно использовать для быстрого создания интерфейсов. Эти компоненты стандартизированы и протестированы, что гарантирует их корректную работу во всех современных браузерах.

**Адаптивность:** Bootstrap использует сеточную систему и медиазапросы для обеспечения корректного отображения страниц на различных устройствах — от мобильных телефонов до настольных компьютеров. Адаптивность позволяет создавать интерфейсы, которые автоматически подстраиваются под размер экрана пользователя, обеспечивая удобство использования на любых устройствах.

**Совместимость и стандарты:** Компоненты фреймворка тестируются и работают во всех современных браузерах. Код написан с соблюдением веб-стандартов, что помогает избежать проблем с отображением и функциональностью. Это позволяет разработчикам сосредоточиться на создании уникального дизайна и функционала, не беспокоясь о кроссбраузерной совместимости.

**Настраиваемость:** Bootstrap легко настраивается под конкретные требования проекта. Разработчики могут использовать SASS-переменные для изменения стандартных стилей, создавая уникальный дизайн. Кроме того, можно включать или отключать отдельные компоненты фреймворка, чтобы оптимизировать производительность приложения.

**Интеграция с JavaScript:** Bootstrap включает в себя множество готовых JavaScript-плагинов, которые позволяют добавлять интерактивные элементы на страницы. Это могут быть всплывающие окна, карусели, выпадающие меню и другие элементы, которые улучшают пользовательский опыт.

Использование Laravel и Bootstrap в одном проекте предоставляет мощный набор инструментов для разработки веб-приложений. Laravel обеспечивает мощный и многофункциональный сервер, предоставляя всю необходимую серверную логику и управление данными, в то время как Bootstrap предоставляет стильный и адаптивный интерфейс для фронтенд-разработки.

**Быстрая разработка:** Laravel и Bootstrap позволяют быстро создавать прототипы и конечные продукты благодаря своим встроенным функциям и компонентам. Использование готовых решений экономит время и ресурсы, позволяя сосредоточиться на уникальных аспектах проекта.

**Чистый и поддерживаемый код:** Архитектура MVC Laravel и структурированность Bootstrap помогают поддерживать код в чистоте и порядке. Это упрощает сопровождение проекта и внесение изменений, делая его более устойчивым к масштабированию и развитию.

**Мощные инструменты разработки:** Laravel предоставляет инструменты для управления базами данных, работы с очередями, кэшированием и другими аспектами серверной логики, а Bootstrap обеспечивает создание адаптивных и привлекательных интерфейсов. Вместе они создают полноценное решение для разработки современных веб-приложений.

**Активное сообщество и поддержка:** Большое сообщество разработчиков Laravel и Bootstrap обеспечивает постоянное обновление и поддержку фреймворков. Это позволяет быть уверенным в актуальности и безопасности используемых технологий.

Laravel и Bootstrap представляют собой мощный тандем для разработки веб-приложений. Laravel предоставляет все необходимые инструменты для создания серверной части приложения, обеспечивая удобную работу с базами данных, маршрутизацией, аутентификацией и другими важными аспектами. Bootstrap, в свою очередь, предлагает обширный набор готовых компонентов и стилей для создания адаптивного и современного интерфейса. В результате, использование этих фреймворков позволяет создавать функциональные, привлекательные и поддерживаемые веб-приложения с минимальными усилиями.

1. **Инсталляция всех необходимых программ и фреймворка, предпроектная подготовка к работе**

Разработка веб-приложения «NBS» на фреймворке Laravel и с использованием Bootstrap требует установки и настройки нескольких программ и инструментов. В этом разделе мы подробно рассмотрим процесс инсталляции всех необходимых компонентов и подготовку к работе.

#### Установка PHPStorm.

PHPStorm будет использоваться как основной редактор кода. Он предоставляет мощные инструменты для разработки на PHP, включая подсветку синтаксиса, автодополнение, отладку и многое другое.

Переходим на официальный сайт [PHPStorm](https://www.jetbrains.com/phpstorm/" \t "C:\\Users\\74BC~1\\AppData\\Local\\Temp\\_new). Скачиваем установочный файл для вашей операционной системы. Установливаем PHPStorm, следуя инструкциям установщика.

**Настройка PHPStorm.**

После установки запустим PHPStorm. Установим необходимые плагины для работы с Laravel и Bootstrap.

#### Установка OpenServer.

OpenServer — это мощный и удобный локальный сервер для Windows, который включает в себя все необходимые компоненты для разработки на PHP. **Скачивание и установка OpenServer: п**ереходим на официальный сайт [OpenServer](https://ospanel.io/" \t "C:\\Users\\74BC~1\\AppData\\Local\\Temp\\_new).Скачиваем последнюю версию OpenServer. Установливаем OpenServer, следуя инструкциям на сайте.

**Настройка OpenServer.**

Запустим OpenServer и выберите необходимую конфигурацию (версии PHP, MySQL и другие компоненты). Перейдите в настройки и укажите корневую папку нашего проекта.

#### Создание репозитория на GitHub.

Для управления версиями проекта и совместной работы создадим репозиторий на GitHub.

**Создание репозитория: п**ереходим на [GitHub](https://github.com/" \t "C:\\Users\\74BC~1\\AppData\\Local\\Temp\\_new) и входим в свой аккаунт. Нажимаем на кнопку "New repository" и создаем новый репозиторий, указав необходимые настройки (имя, описание, доступ). **Настройка локального репозитория: откроем** командную строку (cmd) и перейдем в папку нашего проекта. Выполним команды для инициализации локального репозитория и привязки его к удаленному репозиторию на GitHub:

#### 

Рисунок 2.1.

#### Создание и настройка проекта Laravel.

Теперь установим Laravel и создадим новый проект. **Установка Laravel:**



Рисунок 2.2.

**Создание нового проекта.** Перейдем в папку, где мы хотим создать новый проект, и выполняем команду:



Рисунок 2.3.

**Настройка проекта.** Перейдите в папку нашего проекта:



Рисунок 2.4.

Откройте файл .env и настройте параметры подключения к базе данных:

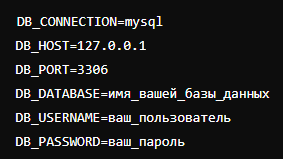


Рисунок 2.5.

**Запуск проекта.** Выполните команду для запуска локального сервера:



Рисунок 2.6.

Откройте браузер и перейдите по адресу http://127.0.0.1:8000, чтобы убедиться, что проект успешно запущен.

#### Установка и настройка Bootstrap.

Для создания адаптивного интерфейса будем использовать Bootstrap.

**Установка Bootstrap.** Откроем командную строку и перейдем в папку нашего проекта.



Рисунок 2.7.

**Настройка Bootstrap.** Откроем файл resources/views/welcome.blade.php и добавим подключение Bootstrap:



Рисунок 2.8.

Теперь у нас настроен проект Laravel с использованием Bootstrap. Можем начинать разработку, используя мощные возможности обоих фреймворков.

### Руководство программиста

Laravel — мощный и гибкий PHP-фреймворк, который упрощает разработку веб-приложений. Он предлагает удобную и логичную структуру каталогов, которая помогает организовать код и делает его поддерживаемым.

#### Основная структура проекта.

Когда мы создали новый проект Laravel, он сам генерирует множество файлов и папок. Ниже приведено описание каждой папки и файла, их функции и взаимодействие:

#### .git

Папка .git содержит все файлы, необходимые для управления версиями с помощью Git. Она скрыта и не требует ручного вмешательства.

#### .idea

Папка .idea используется IDE (например, PhpStorm) для хранения проектных настроек. Она может включать файлы конфигурации, такие как настройки кодировки, версии PHP и другие параметры IDE.

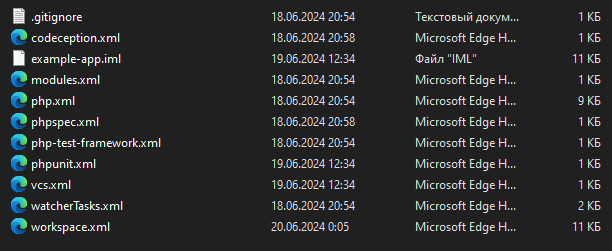


Рисунок 3.1.

#### app

#### Основные подкаталоги и файлы в папке app:

1. **Console**

**-Console/Commands**: Содержит консольные команды, которые можно выполнять через Artisan. Вы можете создать новые команды, расширяя функциональность приложения.

**-Kernel.php**: Определяет расписание выполнения команд и регистрацию команд.

**3) Exceptions**

**-Handler.php**: Содержит логику обработки исключений. Этот файл используется для настройки и управления исключениями, которые происходят в нашем приложении.

**3) Http**

**-Controllers**: Содержит контроллеры, которые обрабатывают HTTP-запросы. Контроллеры отвечают за взаимодействие между моделями и представлениями, определяя логику приложения.

**-Middleware**: Содержит посредники, которые фильтруют HTTP-запросы, проходящие через ваше приложение. Посредники могут выполнять различные задачи, такие как аутентификация и проверка прав доступа.

**-Requests**: Содержит классы запросов, которые используются для валидации данных, поступающих в ваше приложение. Эти классы помогают централизовать и упрощать валидацию входных данных.

1. **Models**

**-User.php**: Является моделью пользователя. По умолчанию, эта модель используется для взаимодействия с таблицей users в базе данных. Модель содержит атрибуты и методы, которые представляют сущности в нашем приложении. Вы можете добавлять методы и свойства, специфичные для логики нашего приложения.

1. **Providers**

**-Providers**: Содержит классы поставщиков сервисов (Service Providers). Поставщики сервисов инициализируют основные компоненты нашего приложения, такие как маршрутизация, события и другие службы.

**-AppServiceProvider.php**: Регистрация общих служб для нашего приложения.

**-AuthServiceProvider.php**: Настройка служб аутентификации и авторизации.

**-EventServiceProvider.php**: Регистрация обработчиков событий.

**-RouteServiceProvider.php**: Настройка маршрутов.

**6)Views**

**-Views**: (этот подкаталог обычно находится в папке resources, но в нашем случае он также присутствует в app) Содержит шаблоны представлений нашего приложения. Представления используются для отображения данных пользователям и являются важной частью MVC-архитектуры. Файлы представлений обычно имеют расширение .blade.php и используют Blade — шаблонизатор Laravel.

Дополнительные файлы и их функции.

**- User.php**: Как упомянуто ранее, этот файл представляет собой модель пользователя. Она использует ORM Eloquent для взаимодействия с базой данных. Модель определяет атрибуты и методы, которые позволяют манипулировать данными пользователей, такими как создание, обновление и удаление записей.

Взаимодействие компонентов в папке app

Папка app играет ключевую роль в структуре нашего приложения, обеспечивая взаимосвязь между различными компонентами:

**-Контроллеры** в папке Http/Controllers обрабатывают входящие HTTP-запросы, используя **модели** из папки Models для взаимодействия с базой данных и **представления** из папки Views для отображения данных.

**-Посредники** в папке Http/Middleware фильтруют запросы и могут выполнять задачи, такие как проверка прав доступа и аутентификация.

**-Запросы** в папке Http/Requests обеспечивают централизованную валидацию входных данных.

**-Поставщики сервисов** в папке Providers инициализируют основные компоненты и службы приложения, обеспечивая корректную работу всех частей системы.

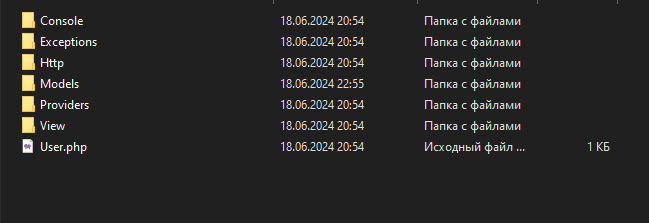


Рисунок 3.2.

#### bootstrap

Папка bootstrap содержит файл app.php, который инициализирует фреймворк. Подкаталог cache используется для хранения скомпилированных файлов для оптимизации производительности.

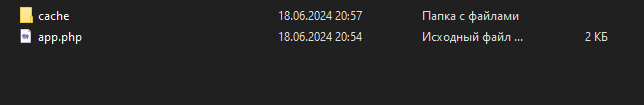


Рисунок 3.3.

#### config

Папка config в Laravel содержит файлы конфигурации, которые позволяют настроить различные аспекты нашего приложения. Эти файлы написаны на PHP и возвращают массивы конфигурационных параметров. Рассмотрим подробнее каждый файл в папке config, а также их назначение и взаимодействие.

Файл app.php содержит общие настройки приложения, такие как название, часовой пояс, локализация и ключ шифрования. Некоторые ключевые параметры:

-name: Имя нашего приложения.

-env: Текущая среда (например, local, production).

-debug: Включение/выключение режима отладки.

-url: Базовый URL нашего приложения.

-timezone: Часовой пояс нашего приложения.

-locale: Язык по умолчанию.

-fallback\_locale: Резервный язык, если основной недоступен.

-key: Ключ шифрования для нашего приложения.

-providers: Список поставщиков сервисов, загружаемых при старте приложения.

-aliases: Список классов-алиасов для быстрого доступа к основным компонентам.

Файл auth.php отвечает за настройки аутентификации. Он определяет guards и providers, используемые для аутентификации пользователей:

-defaults: Устанавливает значения по умолчанию для guard и паролей.

-guards: Определяет, как пользователи будут аутентифицироваться (например, через сессии или токены).

-providers: Указывает, какие источники данных будут использоваться для аутентификации (например, Eloquent-модели или база данных).

Файл broadcast.php содержит настройки для вещания событий. В Laravel встроена поддержка для различных драйверов вещания, таких как Pusher, Redis и Log:

-default: Устанавливает драйвер вещания по умолчанию.

-connections: Содержит параметры для каждого драйвера вещания.

Файл cache.php управляет настройками кэширования нашего приложения:

-default: Устанавливает драйвер кэша по умолчанию (например, file, redis).

-stores: Определяет различные хранилища кэша.

-prefix: Префикс для всех ключей кэша.

Файл cors.php содержит настройки для CORS (Cross-Origin Resource Sharing). Он определяет, какие источники могут отправлять запросы к нашему приложению:

-paths: Пути, для которых применяются правила CORS.

-allowed\_methods: Разрешенные HTTP-методы.

-allowed\_origins: Разрешенные источники.

-allowed\_headers: Разрешенные заголовки.

-exposed\_headers: Заголовки, которые могут быть видны клиенту.

-max\_age: Максимальное время хранения предварительных запросов.

Файл database.php содержит настройки для подключения к базе данных. Laravel поддерживает различные СУБД, такие как MySQL, PostgreSQL, SQLite и SQL Server:

-default: Устанавливает драйвер базы данных по умолчанию.

-connections: Параметры подключения для каждой СУБД.

-migrations: Настройки для миграций базы данных.

-redis: Параметры подключения для Redis.

Файл filesystems.php управляет настройками файловых систем, используемых нашим приложением:

-default: Устанавливает файловую систему по умолчанию.

-disks: Определяет диски для хранения файлов (например, local, s3).

-links: Символические ссылки для различных дисков.

Файл hashing.php содержит настройки для хеширования паролей и других данных:

-driver: Устанавливает драйвер хеширования по умолчанию (например, bcrypt, argon).

-bcrypt: Параметры для bcrypt.

-argon: Параметры для Argon2.

Файл logging.php управляет настройками логирования нашего приложения:

-default: Устанавливает драйвер логирования по умолчанию.

-channels: Определяет каналы логирования (например, stack, single, daily, slack).

Файл mail.php содержит настройки для отправки электронной почты:

-driver: Устанавливает драйвер почты по умолчанию (например, smtp, sendmail).

-host: Хост SMTP-сервера.

-port: Порт SMTP-сервера.

-from: Адрес и имя отправителя по умолчанию.

-encryption: Метод шифрования (например, tls).

-username: Имя пользователя для SMTP.

-password: Пароль для SMTP.

Файл queue.php содержит настройки для очередей задач:

-default: Устанавливает драйвер очереди по умолчанию.

-connections: Определяет подключения для различных драйверов очередей (например, sync, database, redis).

-failed: Настройки для обработки неудачных задач.

Файл sanctum.php содержит настройки для Laravel Sanctum, который обеспечивает простую аутентификацию API с помощью токенов:

-stateful: Определяет домены, которые будут рассматриваться как состояние.

-expiration: Время истечения токенов.

Файл services.php используется для хранения настроек сторонних сервисов, таких как Mailgun, Postmark и другие. Эти настройки включают API-ключи и параметры аутентификации.

Файл session.php управляет настройками сессий нашего приложения:

-driver: Устанавливает драйвер сессий по умолчанию (например, file, cookie, database).

-lifetime: Время жизни сессии.

-expire\_on\_close: Закрытие сессии при закрытии браузера.

-encrypt: Шифрование данных сессии.

Файл view.php содержит настройки для системы представлений нашего приложения:

-рaths: Пути к директориям с представлениями.

-compiled: Путь к директории, где будут храниться скомпилированные шаблоны Blade.

Взаимодействие компонентов

Файлы конфигурации в папке config взаимодействуют с остальными частями нашего приложения, обеспечивая централизованное управление настройками:

**-app.php**: Общие настройки, которые влияют на многие аспекты приложения.

**-auth.php**: Управляет аутентификацией, обеспечивая безопасный доступ к ресурсам.

**-broadcast.php**: Настраивает вещание событий для интерактивных функций в реальном времени.

**-cache.php**: Управляет кэшированием для повышения производительности.

**-cors.php**: Обеспечивает безопасность при взаимодействии с другими доменами.

**-database.php**: Настраивает подключение к базе данных для работы с данными.

**-filesystems.php**: Управляет файловыми системами для хранения и доступа к файлам.

**-hashing.php**: Обеспечивает безопасное хеширование паролей.

**-logging.php**: Настраивает логирование для отслеживания событий и ошибок.

**-mail.php**: Управляет отправкой электронной почты.

**-queue.php**: Настраивает очереди задач для асинхронной обработки.

**-sanctum.php**: Обеспечивает простую аутентификацию API с помощью токенов.

**-services.php**: Управляет настройками сторонних сервисов.

**-session.php**: Управляет сессиями пользователей.

**-view.php**: Настраивает систему представлений.

Эти файлы позволяют вам гибко и легко изменять настройки нашего приложения, делая его более адаптивным и управляемым.

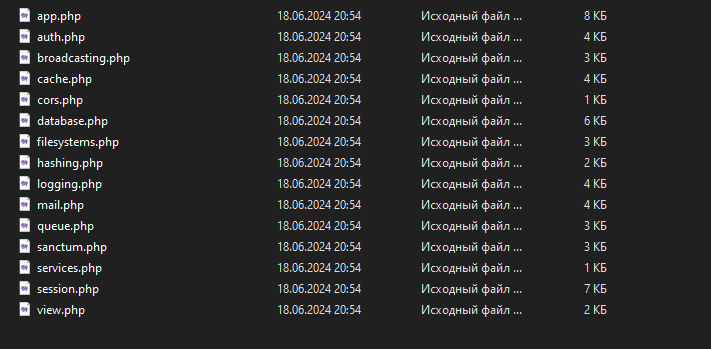


Рисунок 3.4.

#### database

Папка database в Laravel содержит файлы и подкаталоги, которые управляют миграциями, фабриками, посевом данных и другими аспектами работы с базой данных. Эта папка играет ключевую роль в управлении структурой базы данных и данными для нашего приложения.

#### Основные подкаталоги и файлы в папке database

**-factories**

**-migrations**

**-seeders**

**-seeds**

Папка factories содержит фабрики моделей, которые используются для генерации тестовых данных. Фабрики позволяют вам легко создавать экземпляры моделей с предопределенными значениями. Это особенно полезно для тестирования и разработки, когда вам нужно быстро создать множество записей в базе данных.

**-Factory Definition Files**: В этой папке вы найдете файлы фабрик для каждой модели, например, UserFactory.php. В этих файлах определяются атрибуты моделей, которые будут заполнены случайными или предопределенными данными.

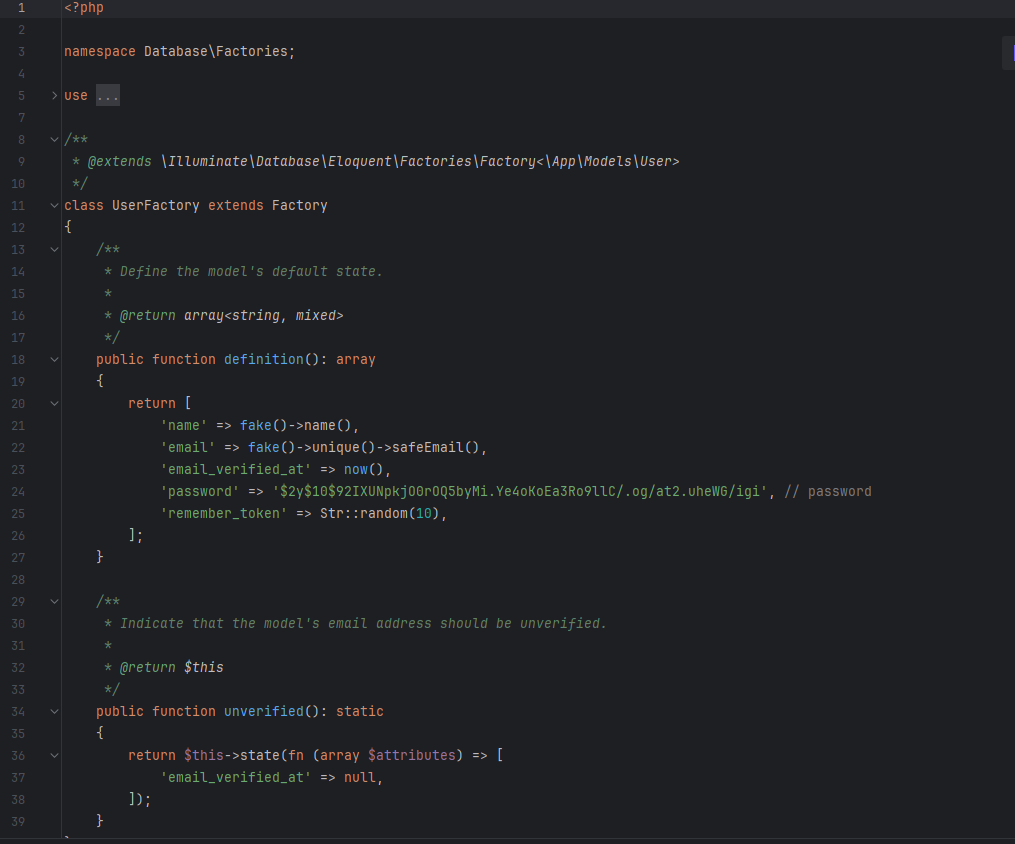


Рисунок 3.5.

Папка migrations содержит файлы миграций, которые управляют структурой вашей базы данных. Миграции позволяют вам создавать, изменять и удалять таблицы и столбцы в базе данных в организованной и последовательной манере. Каждая миграция содержит два метода: up для применения изменений и down для их отмены.

**-Migration Files**: Файлы миграций имеют временные метки в своих именах, что помогает организовать их порядок выполнения.

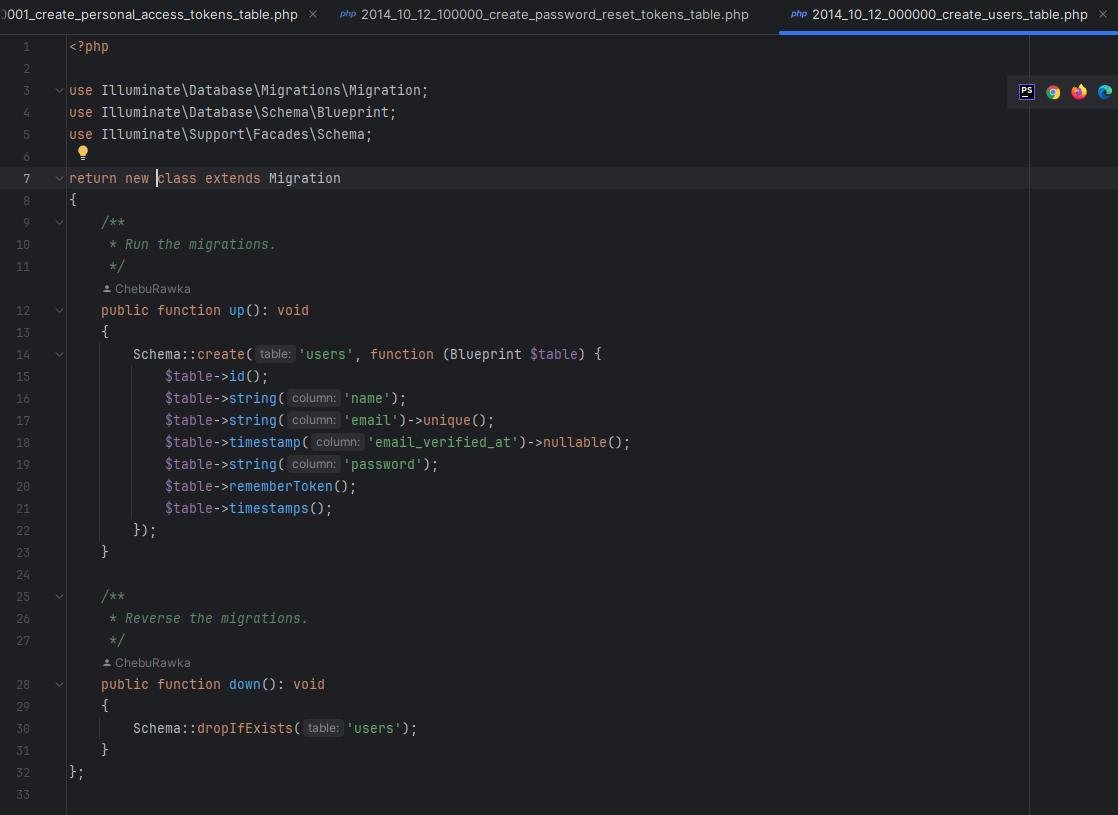


Рисунок 3.6.

Папка seeders содержит классы посева данных, которые позволяют вам заполнять базу данных начальными данными. Эти данные могут быть использованы для тестирования или для предварительного заполнения базы данных при развертывании приложения.

Взаимодействие компонентов в папке database:

**-Фабрики (factories)**: Используются для создания тестовых данных для моделей. Фабрики интегрируются с механизмами тестирования Laravel и позволяют быстро генерировать данные.

**-Миграции (migrations)**: Управляют изменениями структуры базы данных. Они обеспечивают организованное и версионированное применение изменений в базе данных, что делает процесс обновления базы данных безопасным и предсказуемым.

**-Сидеры (seeders)**: Заполняют базу данных начальными данными. Сидеры могут быть вызваны через Artisan команду db:seed или в тестах для установки начального состояния базы данных.

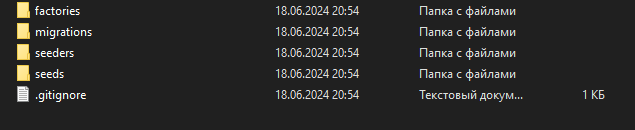


Рисунок 3.7.

**node\_modules**

Папка node\_modules создается менеджером пакетов npm и содержит все зависимости JavaScript, необходимые для моего проекта.

#### public

Папка public в Laravel является корневой директорией, доступной для веб-сервера. Это единственное место, откуда сервер обслуживает файлы нашего приложения. В этой папке находятся все публичные файлы, такие как стили, скрипты, изображения и основной файл входа в приложение.

#### Основные подкаталоги и файлы в папке public

**-image**

**-js**

**-.htaccess**

**-app.css**

**-favicon.ico**

**-hot**

**-index.php**

**-mix-manifest.json**

**-robots.txt**

**-script.js**

**-web.config**

Папка image предназначена для хранения изображений, используемых в нашем приложении. Сюда можно загружать любые изображения, которые будут использоваться на страницах нашего сайта. Примеры использования:

* Логотипы
* Фоновые изображения
* Иконки

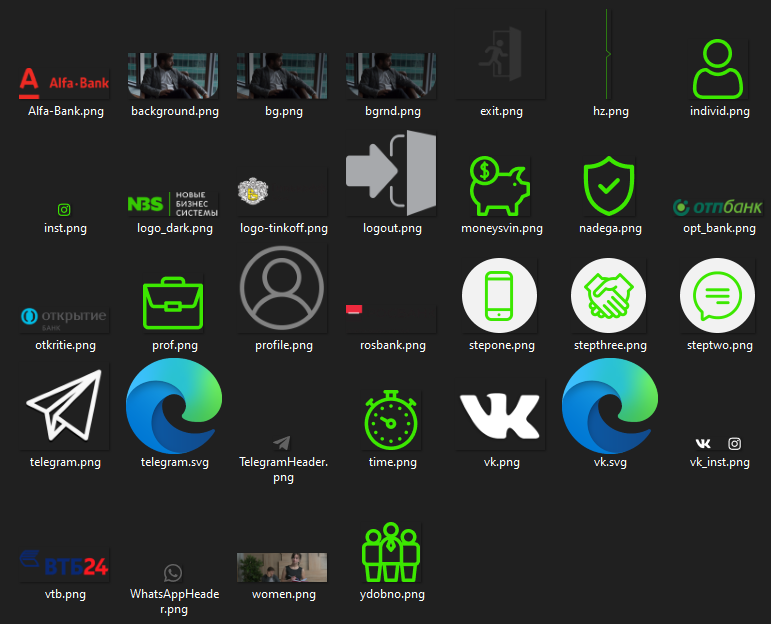


Рисунок 3.8.

Папка js содержит JavaScript-файлы нашего приложения. Здесь вы можете хранить свои собственные скрипты, а также сторонние библиотеки. В типичном проекте Laravel, сюда помещаются файлы, которые вы пишете сами или собираете с помощью инструментов, таких как Laravel Mix.

Файл .htaccess используется для конфигурации веб-сервера Apache. Он позволяет управлять правилами переписывания URL, настройками безопасности и другими параметрами сервера. Пример содержимого .htaccess:

Файл app.css содержит стили нашего приложения. Обычно это файл, скомпилированный из исходных файлов стилей (например, SASS или LESS) с помощью таких инструментов, как Laravel Mix. Этот файл подключается к нашим HTML-страницам для стилизации элементов.

Файл favicon.ico — это иконка нашего сайта, которая отображается в закладках браузера и в заголовке окна. Этот файл должен быть небольшого размера и иметь квадратные пропорции.

Файл hot создается Laravel Mix при запуске приложения в режиме горячей перезагрузки (hot reloading). Он указывает браузеру использовать разработочный сервер для загрузки обновленных ресурсов без необходимости перезагрузки страницы.

Файл index.php является основным входным скриптом нашего приложения. Он инициализирует ядро Laravel и обрабатывает все входящие запросы.

Файл mix-manifest.json создается Laravel Mix и содержит сопоставление между исходными файлами и их скомпилированными версиями. Это позволяет нашему приложению правильно подключать скомпилированные файлы, обеспечивая корректную работу версий и кэширования.

Файл script.js содержит пользовательские JavaScript-коды, которые используются в нашем приложении. Этот файл может включать любые скрипты, которые вы написали для реализации клиентской логики.

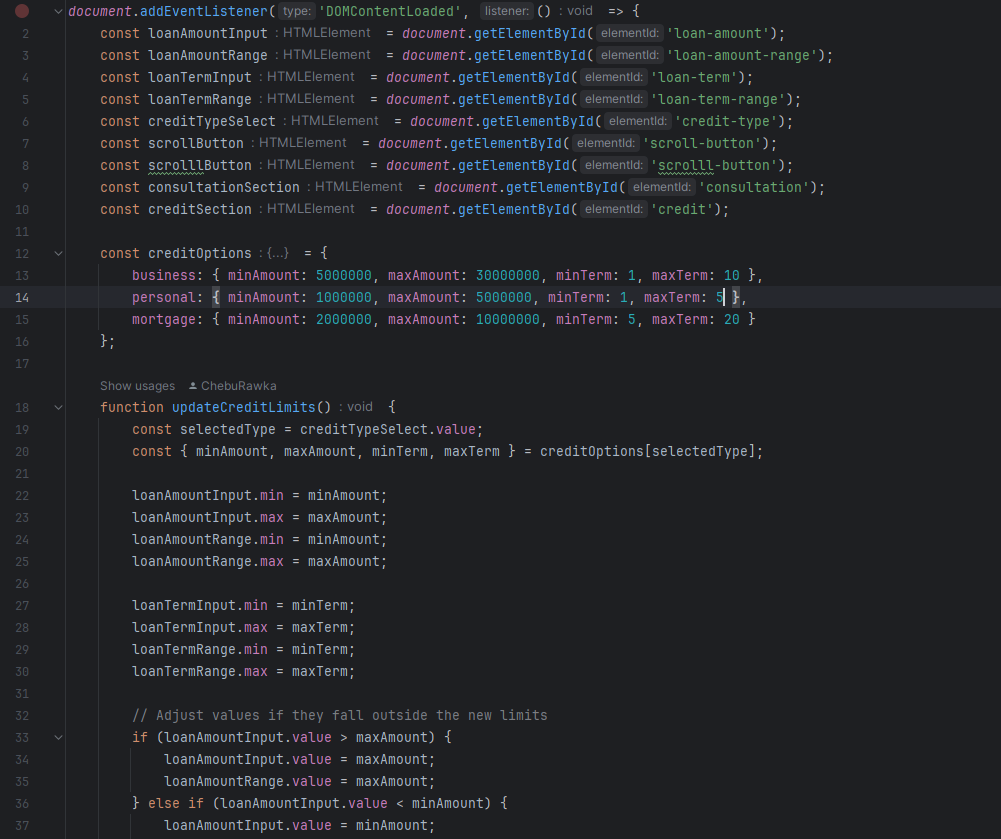


Рисунок 3.9.

Файл web.config используется для конфигурации веб-сервера IIS (Internet Information Services). Этот файл содержит настройки, аналогичные .htaccess, но для серверов Windows.

Взаимодействие компонентов в папке public

**-Изображения (image)**: Изображения загружаются и используются в представлениях (views), чтобы добавить визуальные элементы на страницы нашего приложения.

**-JavaScript (js)**: Скрипты в этой папке подключаются к HTML-страницам для реализации клиентской логики и взаимодействия с пользователем.

**-.htaccess и web.config**: Эти файлы конфигурации управляют доступом к нашему приложению и перенаправлением запросов на index.php.

**-index.php**: Главный входной скрипт, который обрабатывает все входящие запросы и запускает приложение Laravel.

**-app.css и script.js**: Стили и скрипты, подключаемые к нашим HTML-страницам для обеспечения функциональности и внешнего вида.

**-mix-manifest.json**: Помогает правильно подключать скомпилированные ресурсы для обеспечения корректной работы версий и кэширования.

**-favicon.ico**: Иконка сайта, отображаемая в браузере.

**-robots.txt**: Управляет доступом поисковых роботов к нашему сайту.

Эти компоненты работают вместе, чтобы обеспечить правильную работу нашего приложения, предоставляя пользователям доступ к статическим ресурсам, управляющим доступом к серверу и обеспечивая взаимодействие между клиентской и серверной частями приложения.

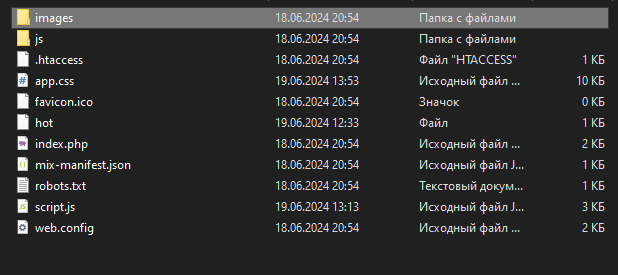


Рисунок 3.10.

#### resources

Папка resources в Laravel предназначена для хранения ресурсов нашего приложения, таких как стили, скрипты, языковые файлы и представления. Эта папка играет важную роль в процессе разработки, предоставляя структурированное место для хранения и управления всеми ресурсами, необходимыми для фронтенда нашего приложения.

#### Основные подкаталоги и файлы в папке resources

**-css**

**-js**

**-lang**

**-sass**

**-views**

Папка css содержит файлы стилей CSS, которые используются в нашем приложении. Эти файлы могут быть исходными стилями или скомпилированными из препроцессоров, таких как SASS или LESS. Обычно, Laravel Mix используется для компиляции и минификации этих файлов перед их размещением в папке public.

Папка js содержит исходные JavaScript-файлы нашего приложения. Эти файлы могут включать в себя собственные скрипты, а также импортированные библиотеки. С помощью инструментов, таких как Laravel Mix, эти файлы могут быть скомпилированы, минифицированы и размещены в папке public.

Папка lang содержит языковые файлы, которые используются для локализации нашего приложения. В Laravel языковые файлы организованы по языковым кодам, таким как en для английского или ru для русского. Внутри каждого подкаталога находятся файлы перевода для различных частей нашего приложения.

Папка sass содержит файлы стилей, написанные на языке SASS (Syntactically Awesome Style Sheets). Эти файлы обычно компилируются в CSS с помощью Laravel Mix и затем размещаются в папке public. Использование SASS позволяет вам писать стили более эффективно и организованно благодаря поддержке переменных, вложенности и миксинов.

Папка views содержит Blade-шаблоны, которые являются представлениями нашего приложения. Blade — это мощный шаблонизатор, встроенный в Laravel, который позволяет писать чистый и удобочитаемый HTML с использованием различных директив и встроенных функций.

Взаимодействие компонентов в папке resources:

**-css и sass**: Файлы стилей, находящиеся в этих папках, обычно компилируются и минифицируются с помощью Laravel Mix и затем перемещаются в папку public для использования в браузере. Это позволяет вам организовывать и писать стили более эффективно.

**-js**: JavaScript-файлы в этой папке также компилируются и минифицируются перед размещением в папке public. Это помогает поддерживать код организованным и уменьшает время загрузки страниц за счет уменьшения размера файлов.

**-lang**: Языковые файлы обеспечивают локализацию нашего приложения, позволяя вам легко добавлять поддержку для нескольких языков. Это особенно полезно для приложений, которые должны обслуживать пользователей из разных регионов.

**-views**: Blade-шаблоны в папке views формируют HTML-страницы нашего приложения. Эти шаблоны могут включать в себя различные части страницы, такие как заголовки, подвал и основной контент, используя директивы Blade для включения, наследования и управления логикой отображения.

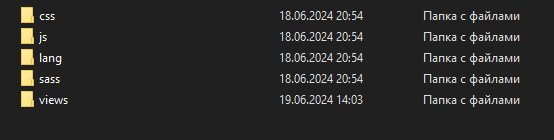


Рисунок 3.11.

#### routes

Папка routes в Laravel содержит файлы, которые определяют маршрутизацию нашего веб-приложения. Маршруты управляют тем, как запросы к нашему приложению обрабатываются и какие контроллеры или функции вызываются для выполнения этих запросов.

#### Основные файлы в папке routes

**-api.php**

**-auth.php**

**-channels.php**

**-console.php**

**-web.php**

Файл api.php используется для определения маршрутов API нашего приложения. Эти маршруты автоматически получают префикс /api и применяют middleware api. Здесь вы можете определять все API-эндпоинты, которые ваше приложение предоставляет.

Файл auth.php предназначен для маршрутов, связанных с аутентификацией. Обычно этот файл используется для определения маршрутов входа, регистрации, сброса пароля и других маршрутов, связанных с управлением учетными записями пользователей.

Файл channels.php используется для определения всех событийных каналов, которые ваше приложение поддерживает. Это важно для использования в реальном времени, таких как уведомления и чаты, где каналы могут быть публичными или приватными.

Файл console.php предназначен для определения всех консольных команд нашего приложения. В этом файле вы можете регистрировать свои собственные Artisan-команды, которые будут выполняться из командной строки.

Файл web.php используется для определения веб-маршрутов нашего приложения. Эти маршруты автоматически получают middleware web, который включает сессии, CSRF-защиту и другие веб-специфические функции. В этом файле вы определяете маршруты, которые обрабатывают запросы к нашему веб-приложению.

Взаимодействие компонентов в папке routes:

api.php: Все маршруты, определенные здесь, предназначены для API-запросов и автоматически получают префикс /api. Эти маршруты обычно возвращают JSON-ответы и предназначены для взаимодействия с клиентскими приложениями через RESTful API.

auth.php: Этот файл обрабатывает все маршруты, связанные с аутентификацией, такие как вход, регистрация и сброс пароля. Это помогает отделить логику аутентификации от остальных маршрутов приложения.

channels.php: Определяет каналы вещания для использования в функциях реального времени, таких как WebSockets. Эти каналы могут быть защищены или публичными в зависимости от требований приложения.

console.php: Содержит определения консольных команд, которые могут быть выполнены через Artisan. Это полезно для создания команд для задач, таких как очистка кеша, управление базой данных и другие административные задачи.

web.php: Основной файл маршрутизации для веб-приложения. Здесь определяются все маршруты, которые обрабатывают HTTP-запросы от браузеров. Эти маршруты используют middleware web для обеспечения защиты и функциональности, таких как сессии и CSRF-защита.

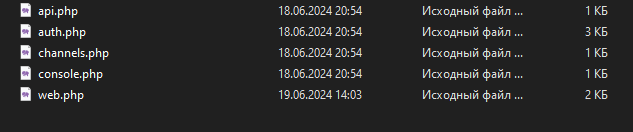


Рисунок 3.12.

#### storage

Папка storage содержит скомпилированные шаблоны, логи и другие файлы, необходимые для работы приложения. Она делится на подкаталоги app, framework и logs.

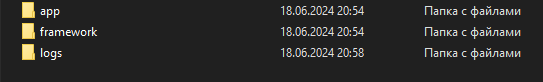


Рисунок 3.13.

#### tests

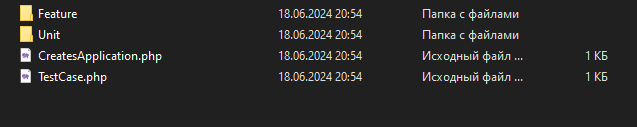
Папка tests содержит тесты нашего приложения. Laravel использует PHPUnit для тестирования. Вы можете создавать тесты для функциональности нашего приложения, чтобы гарантировать его корректную работу.

Рисунок 3.14.

#### vendor

Папка vendor создается Composer и содержит все зависимости PHP, включая сам Laravel и другие библиотеки, которые вы установили через Composer.

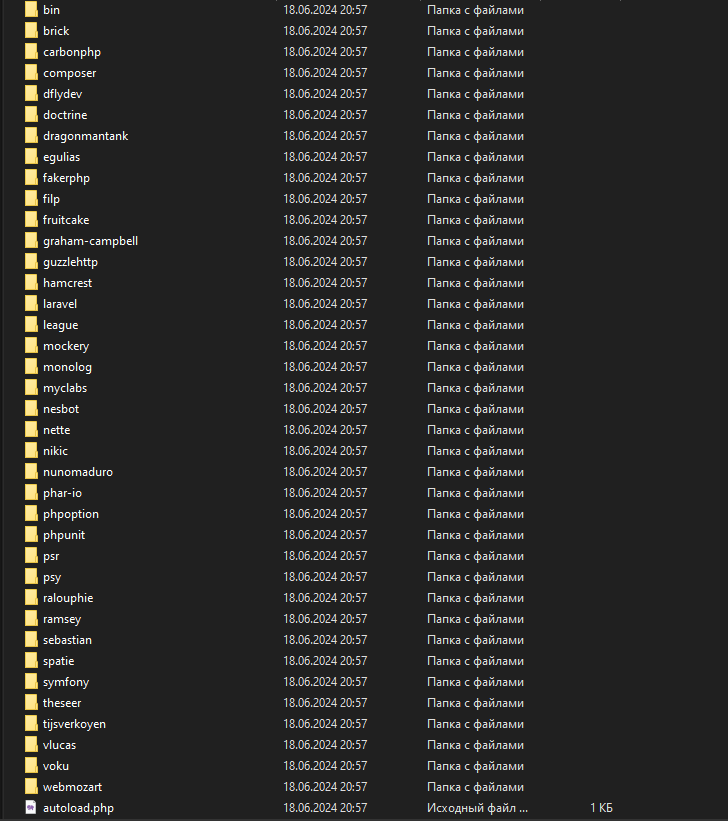


Рисунок 3.15.

#### .env

Файл .env содержит конфиденциальные настройки и параметры конфигурации нашего приложения, такие как настройки базы данных и ключи API. Этот файл не должен быть добавлен в систему контроля версий.

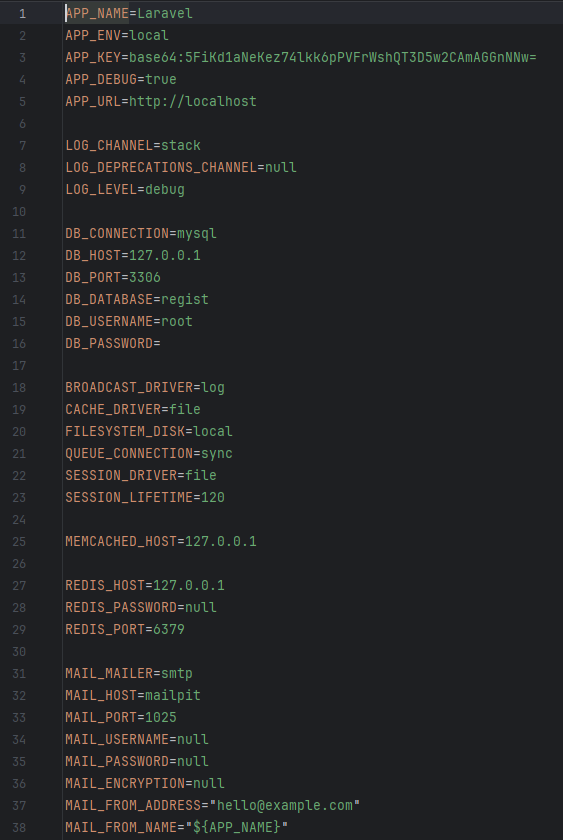


Рисунок 3.16.

#### .gitignore

Файл .gitignore указывает, какие файлы и папки должны быть проигнорированы системой контроля версий Git.

#### artisan

Файл artisan является интерфейсом командной строки для Laravel. Он позволяет выполнять различные команды, такие как миграции базы данных, генерация кода и очистка кэша.

#### composer.json

Файл composer.json содержит метаданные проекта и список зависимостей PHP, используемых нашим приложением. Он также определяет автозагрузку классов.

#### composer.lock

Файл composer.lock фиксирует версии всех установленных зависимостей, чтобы гарантировать, что все разработчики используют одну и ту же версию библиотек.

#### package.json

Файл package.json содержит метаданные проекта и список зависимостей JavaScript. Он используется npm для управления этими зависимостями.

#### package-lock.json

Файл package-lock.json фиксирует версии всех установленных JavaScript-зависимостей, аналогично composer.lock.

#### phpunit.xml

Файл phpunit.xml содержит настройки для PHPUnit, используемого для тестирования нашего приложения.

#### README.md

Файл README.md содержит документацию и описание нашего проекта. Обычно используется для предоставления информации разработчикам и пользователям.

#### server.php

Файл server.php используется встроенным PHP сервером для локальной разработки.

#### vite.config.js

Файл vite.config.js содержит конфигурацию для Vite — современного инструмента сборки для frontend-ресурсов.

#### tailwind.config.js

Файл tailwind.config.js содержит настройки для Tailwind CSS, если вы используете эту библиотеку для стилизации.

Взаимодействие компонентов

Все эти файлы и папки взаимодействуют между собой, чтобы обеспечить корректную работу нашего приложения:

-Запросы поступают в public/index.php, который инициирует Laravel через файлы в bootstrap.

-Контроллеры в app/Http/Controllers обрабатывают запросы, используя данные из моделей в app/Models и возвращают представления из resources/views.

-Конфигурация и параметры окружения загружаются из config и .env.

-Пакеты и библиотеки, установленные через Composer и npm, находятся в vendor и node\_modules соответственно.

1. **Установка веб-приложения на общедоступный хостинг**

Процесс установки Laravel проекта на хостинг Beget

Установка Laravel проекта на хостинг Beget включает несколько шагов, начиная с подготовки локального окружения и заканчивая настройкой базы данных и веб-сервера.

Шаг 1: Подготовка проекта

Проверка зависимостей:

Убедимся, что наш проект Laravel полностью функционален на локальном сервере.

Проверим все зависимости в файле composer.json и выполним команду composer install для установки необходимых пакетов.

Подготовка файловой структуры:

Убедимся, что в корневой папке нашего проекта находятся все необходимые файлы и папки, такие как .env, composer.json, public, routes, app, config, и другие.

Шаг 2: Подготовка хостинга Beget

Создание учетной записи на Beget:

Зарегистрируйтесь и войдите в свою учетную запись на сайте Beget.

Создание нового сайта:

Перейдем в панель управления Beget и создадим новый сайт. Нам будет предложено выбрать доменное имя и папку, в которую будут загружены файлы нашего проекта.

Настройка базы данных:

В панели управления Beget создаем новую базу данных MySQL. Запоминаем имя базы данных, имя пользователя и пароль.

Шаг 3: Загрузка файлов на сервер

Загрузка через FTP:

Подключимся к нашему серверу через FTP. Мы можем использовать такие клиенты, как FileZilla или любой другой FTP-клиент.

Загрузим все файлы и папки нашего Laravel проекта в папку, созданную для нашего сайта на хостинге Beget.

Загрузка через панель управления:

В панели управления Beget мы также можем использовать встроенный файловый менеджер для загрузки файлов нашего проекта.

Шаг 4: Настройка конфигурации

Настройка файла .env:

Обновим файл .env в корне нашего проекта с учетом данных вашей базы данных, предоставленных Beget.

Генерация ключа приложения:

Подключимся к серверу через SSH (если поддерживается нашим тарифом) и выполним команду: php artisan key:generate

Процесс установки Laravel проекта на хостинг Beget

Установка Laravel проекта на хостинг Beget включает несколько шагов, начиная с подготовки локального окружения и заканчивая настройкой базы данных и веб-сервера. В этом руководстве мы рассмотрим все необходимые шаги для успешной установки нашего проекта.

Шаг 1: Подготовка проекта

Проверка зависимостей:

Убедитесь, что ваш проект Laravel полностью функционален на локальном сервере.

Проверьте все зависимости в файле composer.json и выполните команду composer install для установки необходимых пакетов.

Подготовка файловой структуры:

Убедитесь, что в корневой папке нашего проекта находятся все необходимые файлы и папки, такие как .env, composer.json, public, routes, app, config, и другие.

Шаг 2: Подготовка хостинга Beget

Создание учетной записи на Beget:

Зарегистрируйтесь и войдите в свою учетную запись на сайте Beget.

Создание нового сайта:

Перейдите в панель управления Beget и создайте новый сайт. Вам будет предложено выбрать доменное имя и папку, в которую будут загружены файлы нашего проекта.

Настройка базы данных:

В панели управления Beget создайте новую базу данных MySQL. Запомните имя базы данных, имя пользователя и пароль.

Шаг 3: Загрузка файлов на сервер

Загрузка через FTP:

подключимся к нашему серверу через FTP. Вы можете использовать такие клиенты, как FileZilla или любой другой FTP-клиент.

Загрузите все файлы и папки нашего Laravel проекта в папку, созданную для нашего сайта на хостинге Beget.

Загрузка через панель управления:

В панели управления Beget вы также можете использовать встроенный файловый менеджер для загрузки файлов нашего проекта.

Шаг 4: Настройка конфигурации

Настройка файла .env:

Обновите файл .env в корне нашего проекта с учетом данных вашей базы данных, предоставленных Beget.

Генерация ключа приложения:

подключимся к серверу через SSH (если поддерживается нашим тарифом) и выполните команду:

bash

Копировать код

php artisan key:generate

Если SSH недоступен, сгенерируйте ключ на локальном сервере и добавьте его в файл .env.

Шаг 5: Настройка веб-сервера

Настройка корневого каталога:

В панели управления Beget перейдем к настройкам сайта и установим корневую папку в public нашего проекта Laravel.

Настройка файла .htaccess:

Убедимся, что в папке public находится файл .htaccess с правильными правилами перенаправления для Laravel.

Шаг 6: Миграция и заполнение базы данных

Миграции:

Выполним команду миграции на сервере: php artisan migrate --force

Заполнение базы данных:

Выполним команду сидирования: php artisan db:seed --force

Шаг 7: Настройка прав доступа

Настройка прав доступа к папкам:

Убедимся, что папки storage и bootstrap/cache имеют правильные права доступа для записи: chmod -R 775 storage chmod -R 775 bootstrap/cache

Шаг 8: Тестирование и проверка

Проверка работы сайта:

Перейдем по нашему доменному имени и убедитесь, что сайт загружается правильно.

Проверяем основные функции сайта, чтобы убедиться, что всё работает корректно.

Установка Laravel проекта на хостинг Beget требует внимательного следования каждому шагу, чтобы убедиться, что все файлы и конфигурации настроены правильно.

### ЗАКЛЮЧЕНИЕ

В ходе данной работы у меня получилось создать свой проект ‘NBS’ использую фрэймворки Laravel и Bootstrap. Этот процесс включал несколько ключевых этапов, начиная от разработки и настройки проекта на локальном сервере до его успешного развертывания на хостинге Beget. Laravel обеспечил мощную серверную часть, предлагая удобные инструменты для работы с базой данных, маршрутизацией и аутентификацией пользователей. Bootstrap позволил создать удобный пользовательский интерфейс, используя предопределенные стили и компоненты, что значительно ускорило процесс разработки фронтенда.

Работа над проектом ‘NBS’ позволила мне не только ознакомиться с возможностями фреймворков Laravel и Bootstrap, но и углубиться в процесс разработки и развертывания веб-приложений. Благодаря структурированному подходу и детальному рассмотрению каждого этапа, мне удалось создать функциональное и стабильное приложение, готовое к использованию пользователями. Этот проект продемонстрировал важность внимательного планирования и выполнения каждого шага, начиная от начальной настройки среды разработки и заканчивая финальными проверками и тестированием на хостинге. Понимание внутренней архитектуры Laravel и грамотное использование его инструментов позволили обеспечить надежную и эффективную работу приложения.

**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. Лалита М.И., Варвара Р.М., Аслан Х.М. Цифровые технологии в системе управления персоналом / Лалита М.И., Варвара Р.М., Аслан Х.М. [Электронный ресурс] // European Proceedings : [сайт]. — URL: https://www.europeanproceedings.com/article/10.15405/epsbs.2022.12.69 (дата обращения: 26.05.2024).
2. Обзор / [Электронный ресурс] // Django Project : [сайт]. — URL: https://www.djangoproject.com/start/overview/ (дата обращения: 29.05.2024).

# ПРИЛОЖЕНИЕ 2

#### 

Рисунок 2.1.



Рисунок 2.2.



Рисунок 2.3.



Рисунок 2.4.

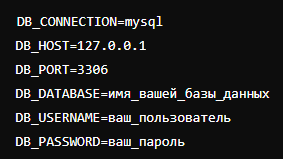


Рисунок 2.5.



Рисунок 2.6.



Рисунок 2.7.



Рисунок 2.8.

# ПРИЛОЖЕНИЕ 3

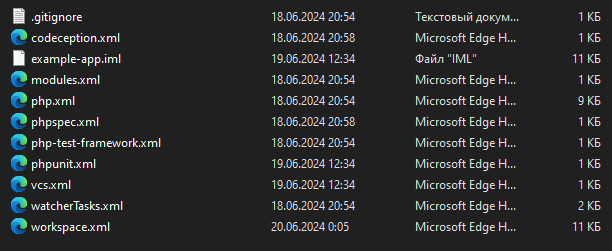


Рисунок 3.1.

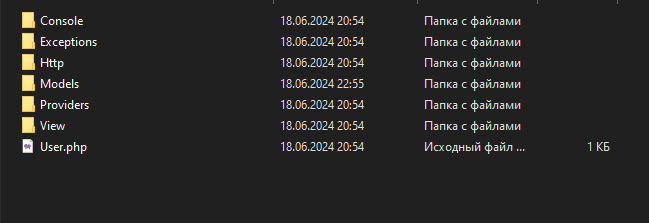


Рисунок 3.2.

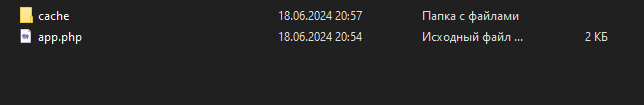


Рисунок 3.3.

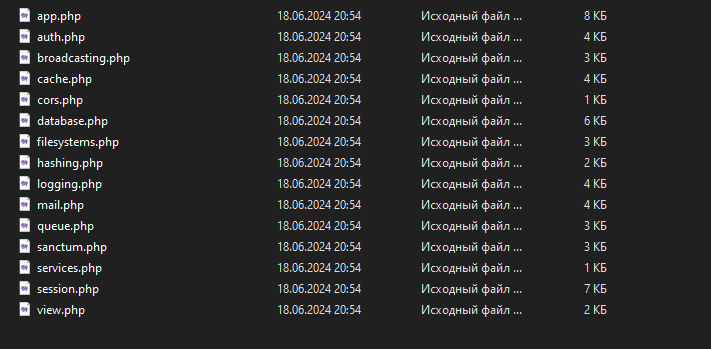


Рисунок 3.4.

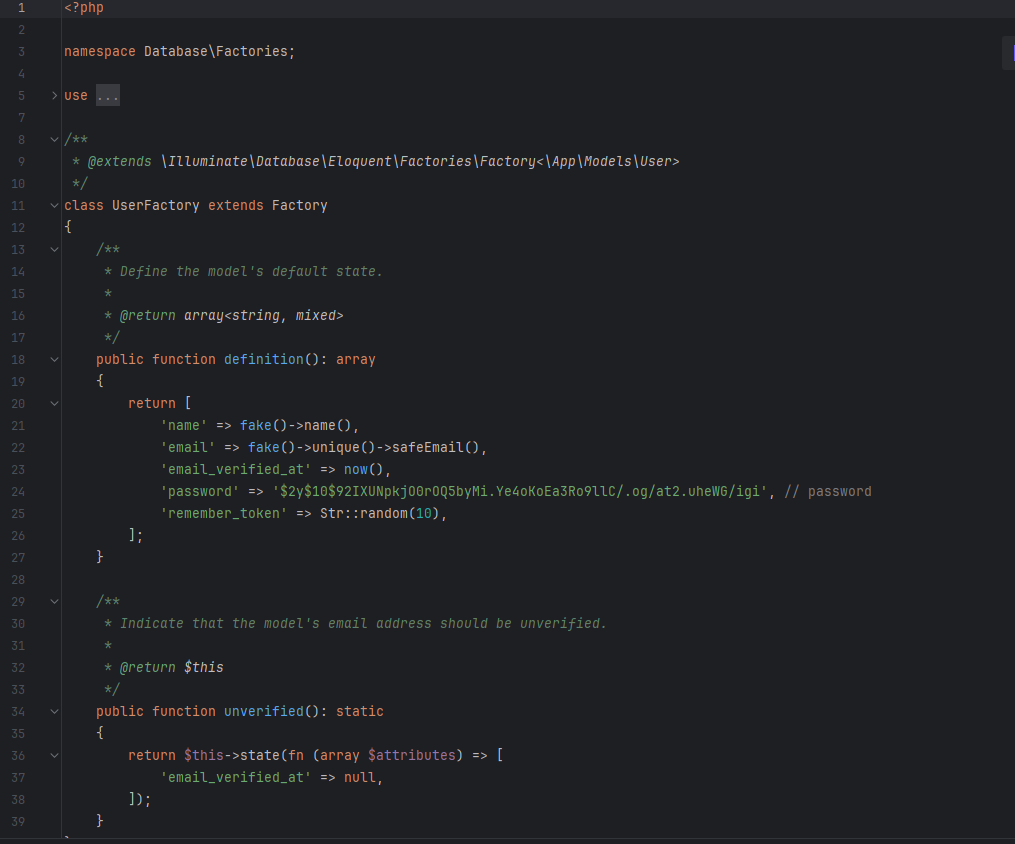


Рисунок 3.5.

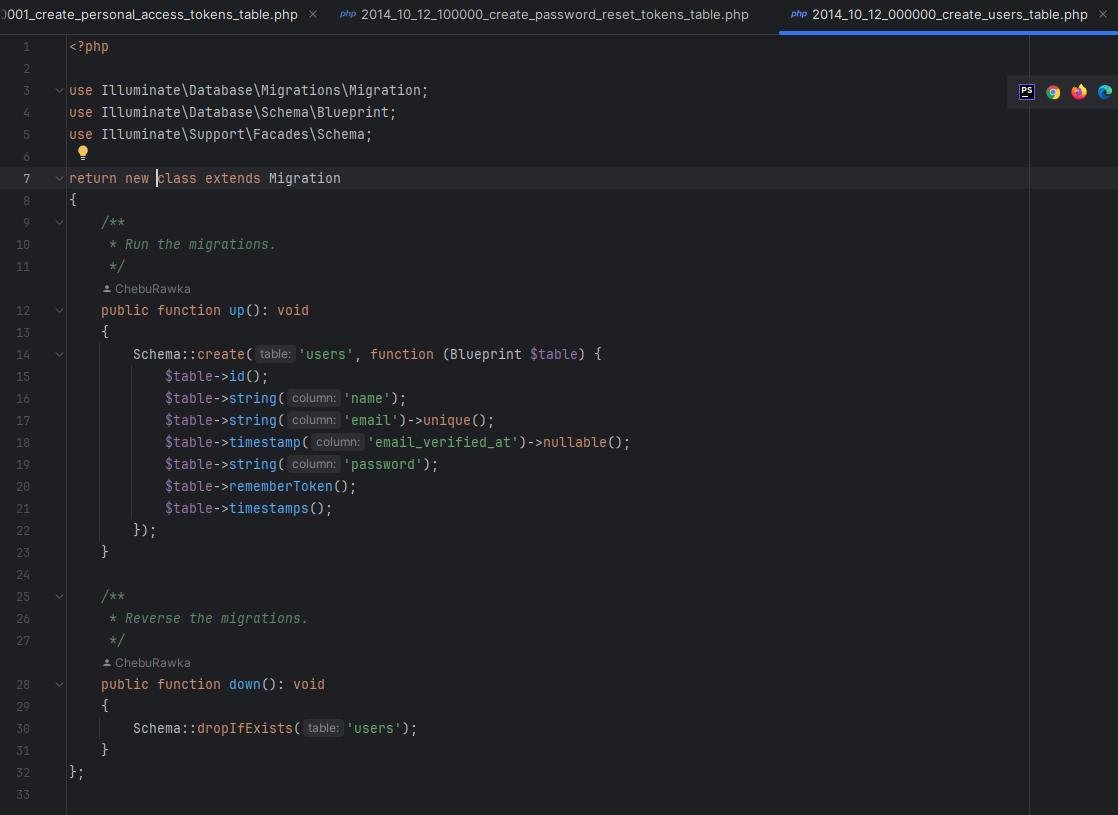


Рисунок 3.6.

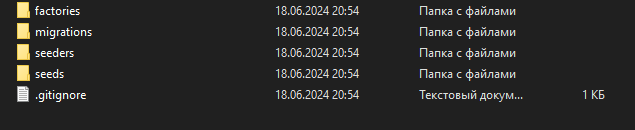


Рисунок 3.7.

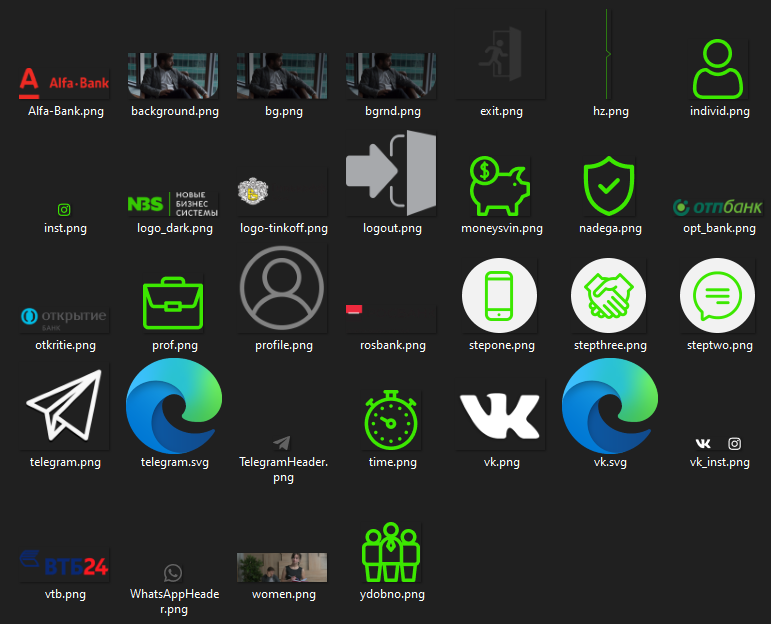


Рисунок 3.8.

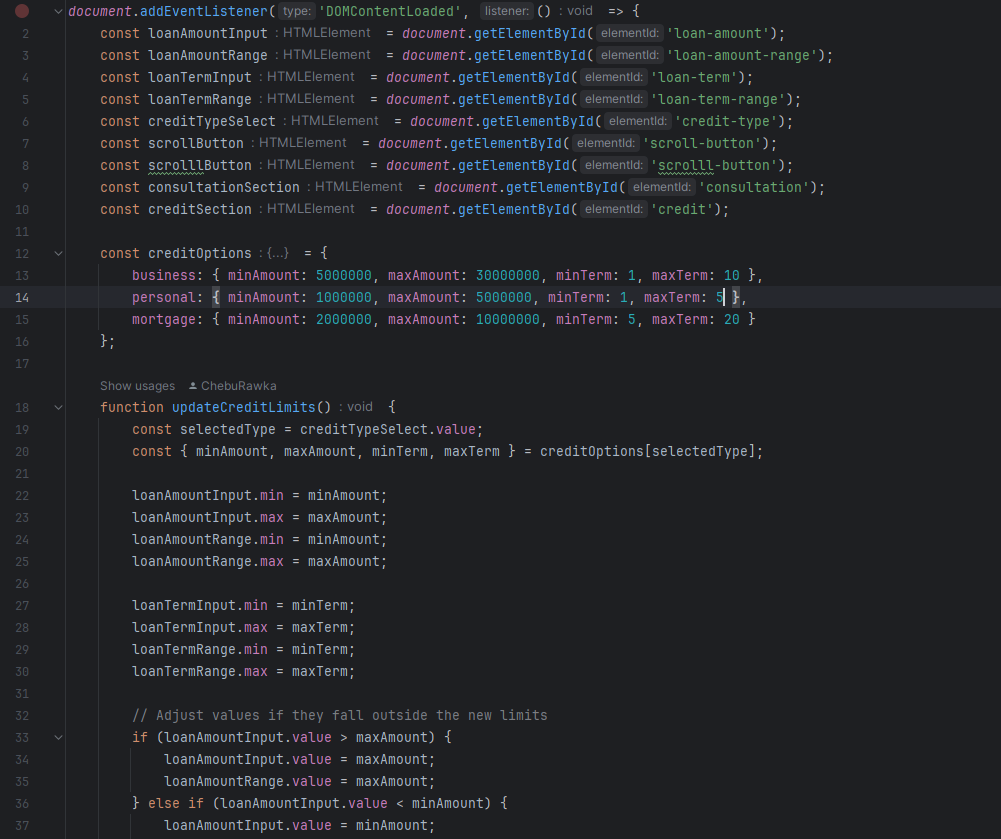


Рисунок 3.9.

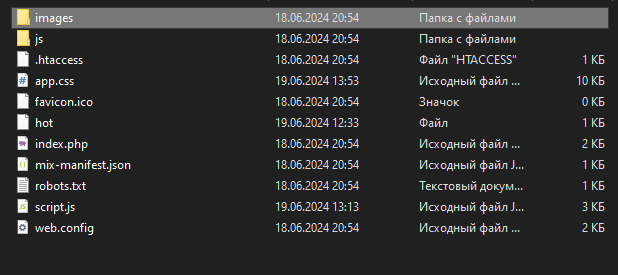


Рисунок 3.10.

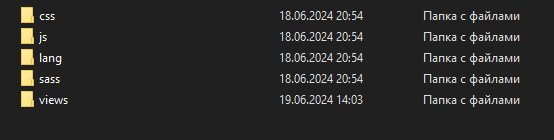


Рисунок 3.11.

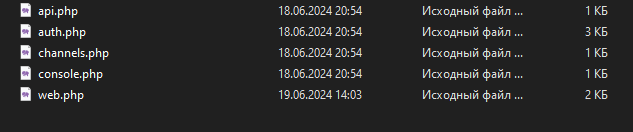


Рисунок 3.12.

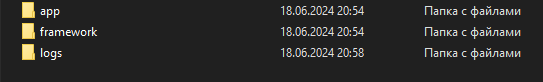


Рисунок 3.13.

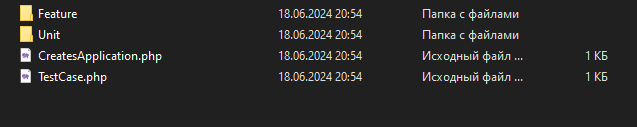


Рисунок 3.14.

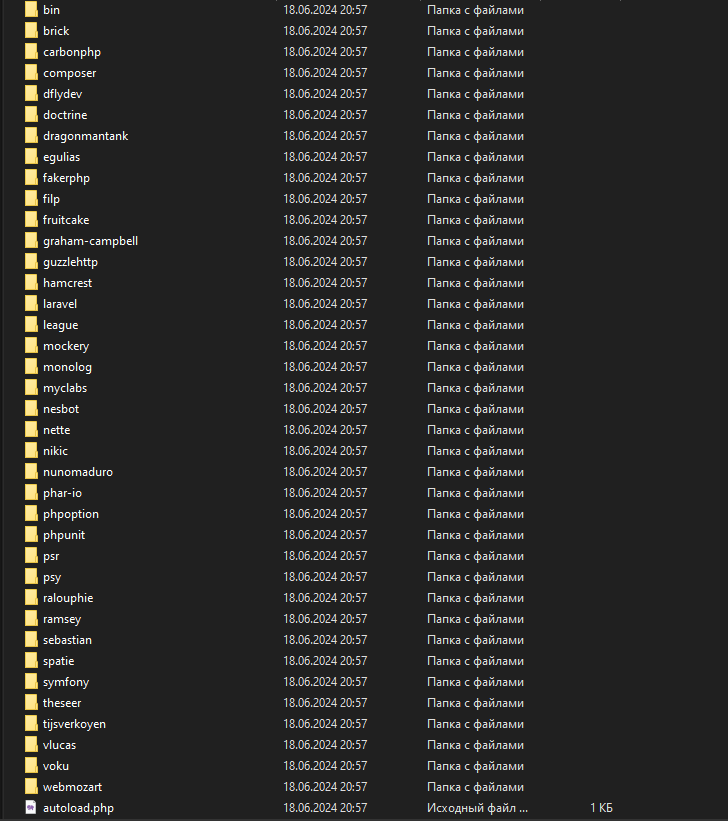


Рисунок 3.15.

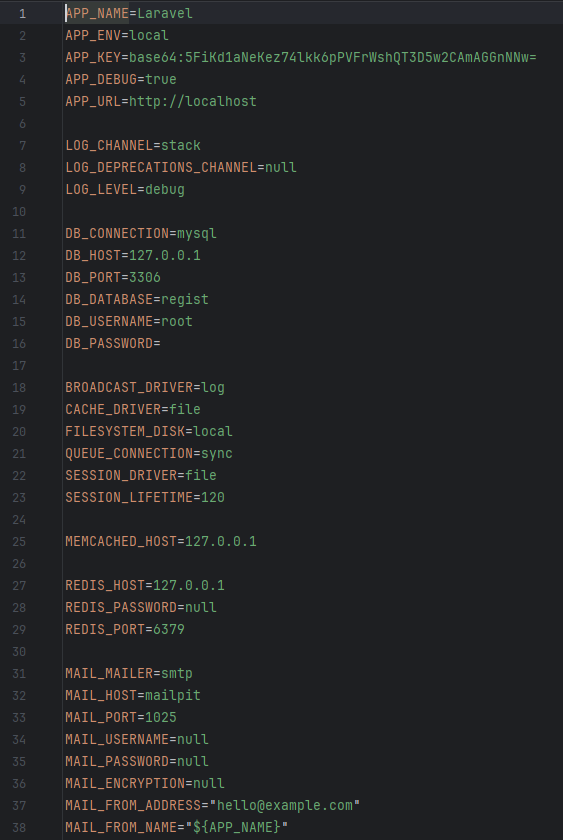


Рисунок 3.16.