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 CAKEPHP В РАЗРАБОТКЕ ВЕБ-ПРИЛОЖЕНИЯ «СОЦИАЛЬНАЯ СЕТЬ»

Разработчик:

студент группы 3ИСПР

Хошвагт Ш.А.

(подпись)

Оценка защиты курсового проекта

Руководитель проекта:

преподаватель

Степанов С.О.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

« »

Дата защиты

« » 2025г.

Ногинск, 2025 г

# Введение

В современном цифровом обществе социальные сети стали неотъемлемой частью повседневной жизни, обеспечивая глобальное взаимодействие, обмен информацией и формирование виртуальных сообществ. Их роль выходит за рамки личного общения, влияя на бизнес, образование и культурные процессы. Разработка таких платформ требует не только понимания потребностей пользователей, но и применения современных технологий, способных обеспечить безопасность, масштабируемость и удобство использования.

Цель курсового проекта заключается в разработке веб-приложения «Социальная сеть».

Для достижения поставленной цели необходимо решить следующие задачи:

* Проанализировать существующие решения на рынке коммуникационных технологий;
* Изучить возможности фреймворка cakephp;
* Провести аналитический обзор литературы.
* Реализовать базовый функционал с использованием фреймворка CakePHP
* Интегрировать базы данных
* Создать пользовательский интерфейс
* Загрузить готовое веб-приложение на хостинг.

Объектом курсового проекта является процесс разработки веб-приложения при помощи программного каркаса (фреймворка PHP) CakePHP.

Предмет курсового проекта: проектирование и реализация веб-приложения «Социальная сеть». Это предполагает реализацию ключевых функций, таких как регистрация и аутентификация пользователей, создание профилей, публикация и управление контентом (текстовые посты, изображения), система подписок и ленты новостей, а также взаимодействие между участниками через комментарии и личные сообщения. Особое внимание уделяется обеспечению безопасности данных, оптимизации запросов к базе данных и созданию адаптивного интерфейса.

Данная тема актуальна на сегодняшний день тем, что социальные сети остаются одним из наиболее востребованных типов веб-приложений, что подчеркивает необходимость изучения современных методов их разработки.

Создание социальной сети как цифрового продукта обладает практической ценностью, актуальной для современных социотехнических систем. Во-первых, платформа способствует формированию виртуальных сообществ, преодолевающих географические и временные ограничения, что критически важно для глобализации знаний, культурного обмена и поддержки профессиональных групп. Во-вторых, социальная сеть выступает инструментом для сбора и анализа больших данных, позволяя исследовать паттерны пользовательского поведения, динамику информационного распространения и социальные связи, что имеет значение для социологии, маркетинга и управления общественными процессами. Проект способствует не только развитию цифровых коммуникаций, но и служит полигоном для междисциплинарных исследований, оптимизации бизнес-моделей и укрепления общественных связей в условиях цифровой трансформации.

# ОСНОВНАЯ ЧАСТЬ

**1.1 Краткая характеристика веб-приложения**

Веб-приложение — это программный продукт, размещаемый на удалённом сервере и распространяемый через интернет посредством браузерного интерфейса. Веб-сервисы, по своей сути, представляют собой подкатегорию веб-приложений, тогда как большинство веб-сайтов, хотя и не все, интегрируют их в свою структуру. Разработка веб-приложений осуществляется для решения разнообразных задач, ориентированных как на корпоративных, так и на частных пользователей. Примерами распространённых веб-приложений являются сервисы электронной почты, онлайн-расчётов, социальных сетей и электронной коммерции. В отличие от классических десктопных программ, требующих установки на устройство, веб-приложения функционируют через браузер, что обеспечивает их доступность на любых гаджетах (смартфонах, планшетах, ПК), независимо от операционной системы. Большинство из них кросс-браузерны, хотя отдельные решения могут ограничивать поддержку определёнными платформами.

Ключевые преимущества веб-приложений:

* Единая версия приложения синхронно доступна пользователям через различные браузеры и устройства (ПК, смартфоны, ноутбуки).
* Минимизация затрат на обслуживание. Отсутствие необходимости локальной установки исключает затраты на поддержку со стороны пользователя. Централизованные обновления обеспечивают актуальность версий и повышенную безопасность.
* Возможность адаптации к растущему числу пользователей или объёмам данных без кардинальных изменений инфраструктуры, что упрощает расширение бизнес-операций.

Социальная сеть – это социальная структура, состоящая из группы узлов, которыми являются социальные объекты (люди, группы людей, сообщества, организации) и связей между ними (социальных взаимоотношений). В общераспространенном понимании социальная сеть – это сообщество людей, объединенных общими интересами, общим делом или имеющих другие причины для непосредственного общения между собой. В общефилософском подходе под социальной сетью понимают множество социальных объектов и определенное множество отношений между ними.

Эллисон (генеральный директор Oracle) и Бойд (microsoft) определили социальную Интернет-сеть как веб-сервис, позволяющий пользователям:

1) создавать открытые (публичные) или частично открытые профили пользователей;

2) создавать список пользователей, с которым они состоят в социальной связи;

3) просматривать и «трассировать» свой список связей и аналогичные списки других пользователей в рамках одной системы.

По мнению Е.Д. Патаркина социальные сети – это платформы, на базе которых участники могут устанавливать отношения друг с другом. Итак, социальная сеть – это интерактивный многопользовательский веб-сайт, обладающий рядом обязательных качеств:

* содержание (контент) сайта создается исключительно или преимущественно его пользователями;
* сайт представляет собой автоматизированную среду, в рамках которой пользователи имеют возможность создавать связи с другими пользователями (социальные связи) или социальные объекты (тематические группы);
* пользователи имеют возможность получать статическую и динамическую информацию об объектах, существующих в данной социальной среде, о социальных связях между ними;
* пользователям доступны функции коммуникации с другими пользователями и социальными объектами.

**1.2 Описание программных инструментов и средств разработки**

1.2.1 Язык программирования «PHP»

PHP (рекурсивный акроним от "PHP: Hypertext Preprocessor") — серверный язык сценариев общего назначения, ориентированный на разработку веб-приложений. PHP выполняется на стороне сервера, генерируя HTML-код, который передается клиенту. Это позволяет создавать динамические сайты, интегрирующиеся с базами данных и внешними API.

Синтаксис PHP сочетает элементы языков C, Java и Perl. Код заключается в теги <?php ... ?>, что обеспечивает встраивание в HTML. Переменные объявляются через символ $, а структуры управления (условия, циклы) используют синтаксис, близкий к C:

<?php

if ($a > $b) {

echo "a больше b";

}

В PHP 7 введена строгая типизация для скалярных типов (int, float, string, bool) с использованием директив declare(strict\_types=1) .

Типы данных включают скалярные (целые, вещественные, строки, булевы), составные (массивы, объекты) и специальные (ресурсы, null). Массивы могут быть ассоциативными, что упрощает работу с данными. В PHP 7 также добавлен тип iterable для итерируемых структур.

Функции поддерживают значения параметров по умолчанию, переменное число аргументов (...$args) и возвращаемые типы:

<?php

function sum(int $a, int $b): int {

return $a + $b;

}

Для веб-разработки PHP предоставляет суперглобальные массивы ($\_GET, $\_POST) для обработки данных форм и управления сессиями через $\_SESSION.

Взаимодействие с базами данных реализуется через расширения mysqli и PDO (PHP Data Objects). PDO поддерживает подготовленные выражения, предотвращающие SQL-инъекции. Пример:

<?php

$stmt = $pdo->prepare("SELECT \* FROM users WHERE id = :id");

$stmt->execute(['id' => $id]);

Безопасность включает валидацию входных данных, экранирование вывода (htmlspecialchars()) и использование HTTPS. Кузнецов и Симдянов (2018) подчеркивают важность хеширования паролей с помощью password\_hash().

В заключение, PHP остается ключевым инструментом для веб-разработки благодаря простоте интеграции с HTML, поддержке СУБД и развитой экосистеме. Эволюция языка, включая нововведения PHP 7, усиливает его производительность и безопасность.

1.2.2 Фреймворк «CakePHP»

Фреймворк (англ. Framework - «каркас», «структура») - это динамически пополняемая библиотека языка программирования, в которой собраны его базовые модули. Фреймворки создаются для упрощения процессов разработки приложений, сайтов, сервисов. Чтобы не писать модуль в приложении с нуля, гораздо проще обратиться к готовым шаблонам фреймворков, которые и формируют рабочую среду разработчика. Архитектура почти всех фреймворков основана на де-композиции нескольких отдельных слоев (приложения, модули и т. д.) проекта. Это означает, что можно расширять функциональность приложения исходя из потребностей и использовать измененную версию вместе с кодом фреймворка или задействовать сторонние приложения. Такая гибкость является одним из одним из ключевых преимуществ использования фреймворков.

Архитектурные принципы и функциональные возможности

CakePHP представляет собой MVC-фреймворк (Model-View-Controller), обеспечивающий структурную основу для веб-приложений. Его архитектура охватывает полный цикл обработки запросов — от инициализации пользовательского взаимодействия до финальной генерации веб-страницы. Следование принципам MVC позволяет кастомизировать и масштабировать компоненты приложения без нарушения целостности системы.

Организационная структура и соглашения

Фреймворк стандартизирует организацию проекта, включая:

* Номенклатуру файлов
* Наименование таблиц базы данных
* Иерархию каталогов

Соблюдение данных конвенций гарантирует логичность архитектуры и упрощает навигацию по кодовой базе. Ключевой принцип: "Следуйте соглашениям — устраните необходимость в избыточной конфигурации".

Минимальные требования CakePHP: веб-сервер и PHP 7.4+. Поскольку базы данных являются неотъемлемой частью большинства веб-приложений, CakePHP поддерживает ряд СУБД, таких как MySQL, PostgreSQL, Microsoft SQL или SQLite (с расширениями PDO).

При соблюдении соглашения об именах CakePHP предоставляет некоторые полезные функции, благодаря которым разработка приложения остается последовательной. К примеру, для написания имен контроллеров используется CamelCase; кроме того, имя контроллера должно быть во множественном числе и заканчиваться словом Controller. Согласно данной договоренности контроллер хранится в одноименном файле.

Имена моделей (Model) должны быть в единственном числе и отражать индивидуальную модель данных. Кроме того, если следовать соглашению, модель будет искать информацию в таблице базы данных с тем же именем в множественном числе и будет использоваться контроллером с тем же именем в множественном числе и со словом Controller в конце.

Составные части веб-приложения на cakephp:

Контроллер – используется для маршрутизации запроса пользователя в браузере к поведению в приложении. Затем он интерпретирует запрос и использует модели, которые должны предоставить запрашиваемую информацию в определенном формате (что определяется видом (Views));

Классы моделей – это слой бизнес логики (business layer) в приложении, они используются для взаимодействия с данными. В CakePHP модели, как правило, представляют собой таблицу базы данных, но они также могут быть использованы для доступа к другим видам данных. В этом отношении модели – это модели данных;

Контроллер используется для маршрутизации запроса пользователя в браузере к поведению в приложении. Затем он интерпретирует запрос и использует модели, которые должны предоставить запрашиваемую информацию в определенном формате;

Представление. Целью представления (View) является отображение данных, запрашиваемых контроллером и предоставленных моделью.

1.2.3 Серверная часть

Операционная система: XUbuntu 18.04 LTS — это дистрибутив Linux на базе Ubuntu, использующий среду рабочего стола XFCE. Выбор обусловлен его стабильностью (долгосрочная поддержка LTS), низким потреблением ресурсов и совместимостью с инструментами разработки. XUbuntu предоставляет минималистичный интерфейс, оптимизированный для серверных и девелоперских задач, что упрощает настройку локального окружения для веб-разработки.

Сборка локального сервера LAMP (Linux, Apache, MySQL, PHP) — стандартный стек для развёртывания веб-приложений. В данной работе используется модификация с заменой MySQL на SQLite3:

* Веб-сервер Apache — высокопроизводительный HTTP-сервер с модульной архитектурой.;
* Интерпретатор языка программирования PHP
  + Xdebug – средство профилирования и отладки PHP-скриптов. Позволяет трассировать выполнение кода, анализировать стек вызовов, измерять время выполнения функций и выявлять узкие места.
  + PHP реализует серверную логику приложения.

Composer – менеджер зависимостей PHP. Упрощает установку библиотек (например, компонентов CakePHP) и управление версиями через файл composer.json. Критичен для поддержания целостности зависимостей в проекте.

Фреймворк CakePHP. MVC-фреймворк с встроенными ORM, механизмами валидации и аутентификации. Ключевые особенности:

* + Конвенция вместо конфигурации (автоматическая маршрутизация по именам контроллеров).
  + Использование шаблонов ActiveRecord и DataMapper для работы с БД.
  + Встроенная защита от CSRF, SQL-инъекций и XSS.

1.2.4 Среда разработки

XFCE Terminal. Эмулятор терминала для среды XFCE. Отличается низким потреблением памяти, поддержкой вкладок и настраиваемыми сочетаниями клавиш. Используется для выполнения CLI-команд (управление Composer, миграциями CakePHP).

SQLite3 – встраиваемая СУБД, хранящая данные в файле. Не требует отдельного серверного процесса, что упрощает развёртывание. Поддерживает транзакции, триггеры и подзапросы.

SQLiteStudio. Графический клиент для SQLite3 с возможностью визуального проектирования схемы БД, написания запросов и экспорта данных. Упрощает отладку структуры таблиц и миграций.

Microsoft Visual Studio Code. Кроссплатформенный редактор с поддержкой расширений. Ключевые функции:

* + Интеграция с Git и терминалом.
  + Подсветка синтаксиса для PHP, SQL, Twig.
  + Отладка через Xdebug (плагин PHP Debug).

CakePHP Snippets. Расширение для VS Code, добавляющее сниппеты (шаблоны кода) для CakePHP. Ускоряет написание контроллеров, моделей и тестов за счёт автоматической генерации заготовок

Unzip. Инструмент для распаковки архивов в формате ZIP. Используется при установке зависимостей Composer или сторонних библиотек, поставляемых в сжатом виде.

# Практическая часть

**2.1 Проектирование веб-приложения**

Функциональные компоненты системы. При проектировании архитектуры социальной сети выделены ключевые функциональные модули, определяющие логику работы приложения:

Модуль регистрации и аутентификации. Реализация поддержки аутентификации, включая регистрацию через электронную почту. Для защиты учетных записей применяется механизм хеширования паролей.

Профили пользователей. Профили содержат базовые данные (имя, аватар, контактная информация) и настраиваемые поля (интересы, статус). Предусмотрена возможность редактирования профиля, настройки видимости данных (приватность) и интеграции медиаконтента.

Управление контентом. Пользователи могут публиковать мультимедийный контент (текст, изображения, видео) с возможностью тегирования и категоризации.

Социальные взаимодействия. Система поддерживает установление связей между пользователями через модель «дружбы» или «подписок». Включает функционал ленты активностей, комментариев, лайков и личных сообщений.

Нефункциональные аспекты проектирования

Архитектура системы разработана с учетом следующих критериев:

Масштабируемость. Применение микросервисной архитектуры и горизонтальное масштабирование (кластеризация с использованием Kubernetes). Использование облачных решений (AWS, Google Cloud) для динамического распределения ресурсов.

Безопасность

Производительность. Оптимизация запросов к базе данных.

CakePHP представляет собой современный фреймворк для разработки веб-приложений, основанный на архитектурном шаблоне Model-View-Controller (MVC). Данный подход обеспечивает структурированное разделение функциональных компонентов приложения, что способствует повышению модульности, поддерживаемости и скорости разработки. В статье рассматриваются ключевые аспекты реализации MVC в CakePHP, включая описание слоёв модели, представления и контроллера, а также типовой цикл обработки запросов.

Архитектура MVC предусматривает разделение приложения на три взаимосвязанных компонента:

* Модель (Model Layer)

Модель реализует бизнес-логику приложения, отвечая за взаимодействие с данными: их извлечение, валидацию, преобразование и хранение. В контексте CakePHP модель выступает абстракцией для работы с базами данных, однако её функции выходят за рамки CRUD-операций. Например, в социальной сети модель может управлять сущностями «Пользователь», «Друг», «Комментарий», обеспечивая обработку связей между объектами, генерацию рекомендаций и т.д. Таким образом, модель инкапсулирует ключевые концепции предметной области.

* Представление (View Layer)

Слой представления ответственен за визуализацию данных, полученных от модели, в формате, приемлемом для клиента (HTML, XML, JSON и др.). Он не ограничивается генерацией HTML-страниц: возможен вывод мультимедийного контента, документов или специализированных форматов. Представление обеспечивает декомпозицию интерфейса, позволяя независимо модифицировать визуальные компоненты без изменения бизнес-логики.

* Контроллер (Controller Layer)

Контроллер обрабатывает пользовательские запросы, координируя взаимодействие модели и представления. Его функции включают:

* Проверку авторизации и аутентификации.
* Делегирование запросов к модели для обработки данных.
* Выбор типа представления в соответствии с требованиями клиента.
* Инициацию рендеринга результата через слой представления.

Контроллер действует как посредник, обеспечивая соответствие workflow приложения заданным бизнес-правилам.

Типичный жизненный цикл запроса в CakePHP включает следующие этапы (Рис. 1):

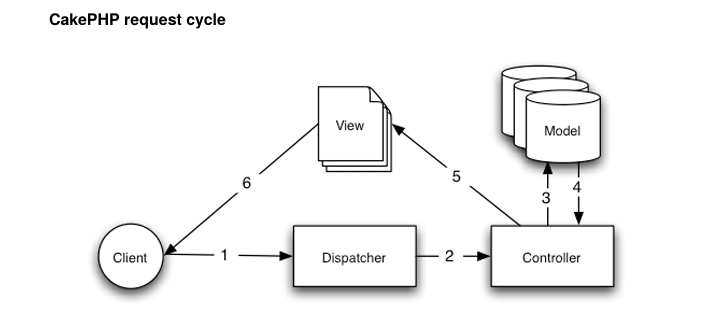


Рисунок 1. Работа веб-приложения

* Роутинг запроса: Диспетчер направляет входящий запрос соответствующему контроллеру.
* Обработка в контроллере: Контроллер взаимодействует с моделью для выполнения операций с данными (извлечение, сохранение).
* Генерация представления: Контроллер передаёт данные в представление, которое формирует ответ в требуемом формате.
* Отправка результата: Сгенерированный вывод возвращается пользователю.

Данный процесс унифицирован для большинства операций, что упрощает масштабирование и поддержку приложения.

Применение шаблона MVC в CakePHP обеспечивает ряд преимуществ:

* Модульность: Чёткое разделение компонентов облегчает повторное использование кода.
* Поддерживаемость: Изменения в одном слое минимально влияют на другие.
* Параллельная разработка интерфейса и бизнес-логики ускоряет создание прототипов.
* Гибкость: Поддержка множества форматов вывода и лёгкая интеграция внешних сервисов.

**2.2 Подготовка к работе**

**Установка интегрированной среды разработки Visual Studio Code в операционной системе Ubuntu**

Процесс инсталляции Visual Studio Code (далее - VS Code) в среде Ubuntu предусматривает несколько методов, основным из которых является подключение официального репозитория Microsoft. Для реализации данного подхода требуется выполнение следующей последовательности действий:

* Установка обязательных зависимостей. Предварительно необходимо обеспечить наличие пакетов, обеспечивающих работу с репозиториями:
  + sudo apt install software-properties-common apt-transport-https wget
* Импорт GPG-ключа репозитория. Аутентичность пакетов подтверждается добавлением криптографического ключа:
  + wget -q https://packages.microsoft.com/keys/microsoft.asc -O- | sudo apt-key add -
* Регистрация репозитория в системе. Добавление источника пакетов с указанием архитектуры (amd64) и ветки стабильных версий:
  + sudo add-apt-repository "deb [arch=amd64] https://packages.microsoft.com/repos/vscode stable main"
* Обновление пакетного индекса и установка приложения. После внесения изменений в конфигурацию требуется синхронизировать данные репозиториев:
  + sudo apt-get update
  + sudo apt-get install code

Альтернативные методы установки:

* Использование менеджера snap-пакетов (требует права root):
  + - * sudo snap install --classic code
    - Инсталляция через графический интерфейс Ubuntu Software Center (поиск по запросу "Visual Studio Code").

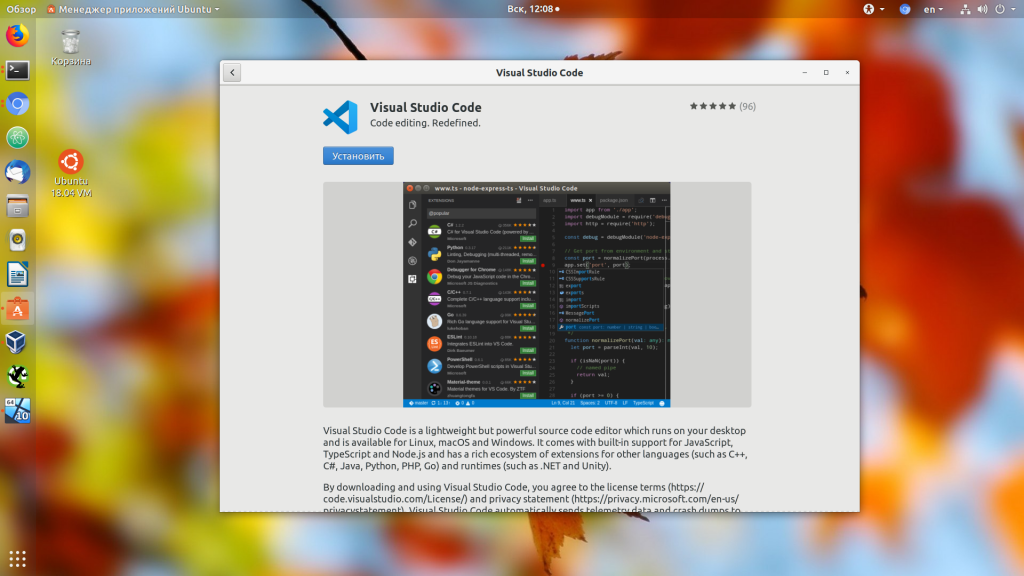


Рисунок 2. Установка VSC

**Установка системы управления зависимостями Composer в операционной системе Ubuntu**

Процедура инсталляции Composer в Ubuntu требует предварительной подготовки системы, так как пакет отсутствует в официальных репозиториях. Алгоритм установки включает два варианта: локальная (в рамках проекта) и глобальная (системная) инсталляция.

* Подготовка системы. Предварительно требуется выполнить обновление системы и инсталляцию необходимых компонентов:
* sudo apt update
* sudo apt install curl php-cli php-mbstring git unzip

Локальная установка (в рамках проекта)

* Переход в рабочую директорию проекта:
  + cd /путь/к/проекту/
* Загрузка установочного скрипта:
  + curl -sS https://getcomposer.org/installer -o composer-setup.php
* Генерация исполняемого файла composer.phar:
  + php composer-setup.php
* Верификация работоспособности:
  + php composer.phar
* Глобальная установка (системный уровень)
  + Загрузка установочного скрипта:
    - curl -s https://getcomposer.org/installer -o composer-setup.php
  + Инсталляция в системную директорию:
    - sudo php composer-setup.php --install-dir=/usr/local/bin --filename=composer
  + Проверка корректности установки:
    - composer
  + Работа с системой управления зависимостями
    - Инициализация конфигурационного файла. Для управления зависимостями проекта используется файл composer.json, создаваемый командой:
      * composer init

В процессе инициализации пользователь может принять значения по умолчанию или указать параметры проекта (название, описание, лицензия, зависимости).

Структура файла composer.json

Файл содержит декларативное описание зависимостей в формате JSON:

{

"require": {

"vendor/package": "version"

},

"require-dev": {

"vendor/package": "version"

}

}

require — обязательные зависимости для выполнения проекта.

require-dev — зависимости, необходимые только на этапе разработки.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Установка фреймворка CakePHP на виртуальный выделенный сервер: методика и этапы

Для развертывания фреймворка CakePHP на виртуальном выделенном сервере доступны несколько методов:

* Загрузка архива с официального сайта.
* Клонирование репозитория через Git.
* Использование менеджера пакетов PEAR.

Наиболее предпочтительным является первый метод, характеризующийся минимальной сложностью и отсутствием зависимости от дополнительных инструментов.

Процедура установки

* Загрузка дистрибутива. Последняя стабильная версия фреймворка загружается с официального сайта CakePHP. Архивный файл размещается в корневом каталоге веб-сервера (/var/www/) для последующей распаковки.
* Установка системных зависимостей
  + Для работы с архивами ZIP требуется пакет unzip. Установка выполняется командой:
    - sudo apt-get install unzip
* Распаковка дистрибутива. Архив распаковывается с сохранением структуры файлов:
  + unzip 2.3.9.zip
  + Примечание: версия архива должна соответствовать актуальному релизу.
* Реорганизация каталогов
  + Созданная директория переименовывается для удобства:
  + mv cakephp-cakephp-4b9e390 /var/www/project
* Настройка прав доступа
  + Для корректной работы фреймворка требуется обеспечить права записи для веб-сервера в каталог app/tmp:
    - cd /var/www/project
    - chown -R root:www-data app/tmp
    - chmod -R 775 app/tmp
  + Обоснование:
  + Команда chown делегирует права владения пользователю root и группе www-data, к которой относится процесс Apache.
  + Режим 775 разрешает чтение, запись и выполнение для владельца и группы, ограничивая права остальных пользователей.

**2.3 Разработка веб-приложения**

Создание проекта cakephp

* Установка Composer, если Composer не установлен:
  + sudo apt update
  + sudo apt install composer
* Создание проекта CakePHP. Выполните команду:
  + composer create-project --prefer-dist cakephp/app social
* Переход в директорию проекта
  + cd social
* Запустить встроенную проверку:
  + bin/cake server
* Остановить сервер (Ctrl+C) и исправить ошибки, если они выводятся

Создание базы данных.

* Переход в директорию config/schema/
  + cd config/schema/
* Запустить sqlite
  + Sqlite3
* Создать новую базу данных
  + .open users.db
* Создать таблицу users
  + CREATE TABLE users (id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT, surname TEXT, lastname TEXT, birthday TEXT, tag TEXT, age INTEGER);

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Принимаемое значение | Тип данных принимаемого значения | Свойства |
| id | Номер пользователя | INTEGER  (целочисленное значение) | PRIMARY KEY  (Ключевое) |
| AUTOINCREMENT  (Автозаполнение, счётчик) |
| name | Имя | TEXT |  |
| surname | Фамилия | TEXT |  |
| lastname | Отчество | TEXT |  |
| birthday | Дата рождения | DATE |  |
| tag | Имя пользователя | TEXT |  |
| age | Возраст | INTEGER  (целочисленное значение) |  |

Таблица 1. Список строк в базе данных «Пользователи»

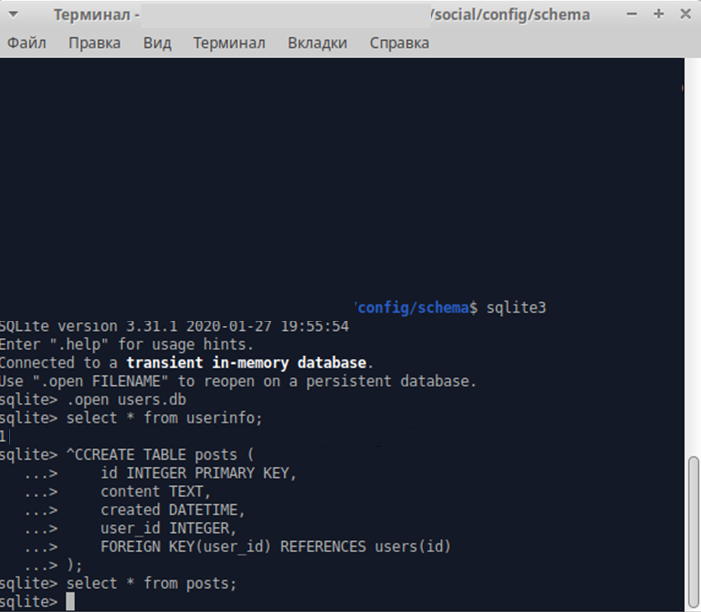


Рисунок 3. Создание базы данных

Подключение базы данных и установка параметров безопасности (хэширование пароля).

* Переход в директорию config/
  + cd config/
* Открыть файл настроек app\_local.php
* Установка «соли»
  + 'Security' => [
  + 'salt' => env('SECURITY\_SALT', 'salt'),
  + ],
* Подключение к базе данных. Используем модуль SQLite.
  + 'Datasources' => [
  + 'default' => [
  + 'className' => 'Cake\Database\Connection',
  + 'driver' => 'Cake\Database\Driver\Sqlite',
  + 'database' => CONFIG. 'schema/users.db',
  + ],

Для валидации (проверки данных) контроллерам, и для вывода данных в формы, необходимо создать модель.

* + Переход в директорию src/Model/Table
    - cd src/Model/Table
* Открыть файл настроек UsersTable.php
* Связь с базой данных

<?php

namespace App\Model\Table;

use Cake\ORM\Table;

class UsersTable extends Table {

public function initialize(array $config): void {

$this->setTable('users');

}

}

Реализация функционала «Регистрация пользователей».

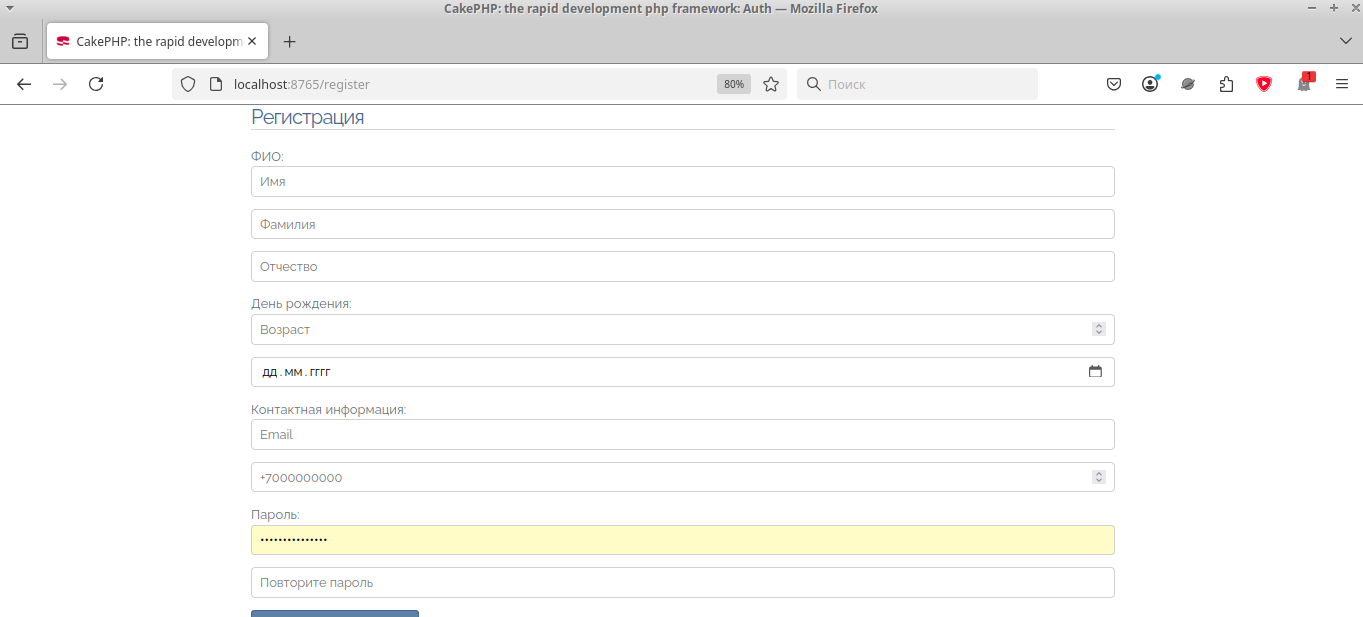


Рисунок 4. Форма регистрации.

Компонент авторизации-регистрации является неотъемлемой частью логики веб-приложения. Без авторизации приложение не сможет идентифицировать пользователя, связать профиль с личностью пользователя, связать интересы, сессию, посты с профилем, без регистрации же не возможно создание профиля, авторизация.

Форма регистрации содержит в себе такие поля, как имя пользователя, пароль, повторение пароля:

|  |  |  |
| --- | --- | --- |
| **placeholder** | **type** | **name** |
| Имя | text | name |
| Фамилия | surname |
| Отчество | lastname |
| Возраст | number | age |
| День рождения | date | birthday |
| Почта | text | email |
| Телефон | number | phone |
| Пароль | password | password |
| Повторения пароля | password | currentpassword |

Таблица 2. Поля формы регистрации.

В представлении (templates/Auth/register.php) необходимо поместить форму регистрации:

<h1 class="registration-title">Регистрация</h1>

<form method="post">

<p class="registration-labels">ФИО:</p>

<input type="text" name="name" placeholder="Имя" required>

<input type="text" name="surname" placeholder="Фамилия" required>

<input type="text" name="lastname" placeholder="Отчество" required>

<p class="registration-labels">День рождения:</p>

<input type="number" name="age" placeholder="Возраст" required>

<input type="date" name="birthday" placeholder="ДР.ММ.ГГ" required>

<p class="registration-labels">Контактная информация:</p>

<input type="email" name="email" placeholder="Email" required>

<input type="number" name="phone" placeholder="+7000000000" required>

<p class="registration-labels">Пароль:</p>

<input type="password" name="password" placeholder="Пароль" required>

<input type="password" name="confirm\_password" placeholder="Повторите пароль" required>

<button type="submit">Зарегистрироваться</button>

</form>

Возраст пользователя должен быть вычислен автоматически: пользователь вводит дату рождения, а приложение высчитывает разницу между сегодняшним днём и днем рождения регистрируемого пользователя и заполняет поле «возраст» числовым значением. Для этого необходимо добавить следующий скрипт:

<script>

document.querySelector('input[name="birthday"]').addEventListener('change', function() {

const birthday = new Date(this.value);

const today = new Date();

let age = today.getFullYear() - birthday.getFullYear();

const monthDiff = today.getMonth() - birthday.getMonth();

const dayDiff = today.getDate() - birthday.getDate();

if (monthDiff < 0 || (monthDiff === 0 && dayDiff < 0)) {

age--;

}

document.querySelector('input[name="age"]').value = Math.max(age, 0);

});

</script>

Необходимо реализовать бэкенд для формы регистрации: обработку принимаемых данных с формы.

Реализация контроллера авторизации регистрации.

Класс: App\Controller\AuthController

Назначение: Реализация механизмов аутентификации, регистрации и восстановления пароля пользователей.

* Переход в директорию src/Contoller/
  + cd src/controller
* Создать файл AuthController.php

<?php

namespace App\Controller;

use App\Controller\AppController;

class AuthController extends AppController {

public function initialize(): void

{

parent::initialize();

$this->loadModel('Users');

}

public function login() {

if ($this->request->is('post')) {

$email = $this->request->getData('email');

$password = $this->request->getData('password');

if (empty($email)) {

$this->Flash->error('Введите email');

return;

}

$user = $this->Users->find()

->where(['email' => $email])

->first();

if ($user && password\_verify($password, $user->password)) {

if ($user->blocked !=1){

$session = $this->request->getSession();

$session->write([

'user\_id' => $user->id,

'user\_email' => $user->email

]);

return $this->redirect(['controller' => 'Users', 'action' => 'home/']);}

else{

$this->Flash->error('Вы заблокированы');

}

} else {

$this->Flash->error('Неверные данные');

}

}

}

public function register() {

if ($this->request->is('post')) {

$data = $this->request->getData();

if ($data['password'] !== $data['confirm\_password']) {

$this->Flash->error(\_\_('Пароли не совпадают'));

return;

}

$usersTable = $this->getTableLocator()->get('Users');

$user = $usersTable->newEmptyEntity();

$userData = [

'email' => $data['email'],

'password' => password\_hash($data['password'], PASSWORD\_DEFAULT),

'name' => $data['name'],

'surname' => $data['surname'],

'birthday' => $data['birthday'],

'phone' => $data['phone'],

'access\_rights' => 0,

'created' => date('Y-m-d H:i:s'),

'modified' => date('Y-m-d H:i:s')

];

$user = $usersTable->patchEntity($user, $userData);

if ($usersTable->save($user)) {

$this->Flash->success(\_\_('успешная регистрация'));

return $this->redirect(['controller' => 'Users', 'action' => 'home']);

}

$this->Flash->error(\_\_('Registration error: {0}', implode(', ', $user->getErrors())));

}

}

public function recovery()

{

if ($this->request->is('post')) {

$data = $this->request->getData();

$user = $this->Users->find()

->where([

'email' => $data['email'],

'phone' => $data['phone']

])

->first();

if ($user) {

$user = $this->Users->patchEntity($user, [

'password' => $data['new\_password']

]);

if ($this->Users->save($user)) {

$this->Flash->success('Пароль успешно изменен');

return $this->redirect(['action' => 'login']);

}

$this->Flash->error('Ошибка изменения пароля');

} else {

$this->Flash->error('Неверные email или телефон');

}

}

}

}

Метод initialize инициализирует контроллера. Выполняет наследование родительского метода initialize() для базовой настройки контроллера.

Подключает модель Users через метод loadModel(), обеспечивая взаимодействие с таблицей пользователей в БД.

Метод register() для регистрации нового пользователя.

* Проверка типа запроса: Активация только для HTTP POST.
* Сравнение полей password и confirm\_password (ошибка при несовпадении).
* Создание пользователя:
  + Генерация хеша пароля через password\_hash()
  + Формирование объекта user с полями: email, password (хеш), name, surname, birthday, phone, access\_rights=0.

Для создания, редактирования, и обработки данных необходимо связывать контроллер с соответствующей таблицей базы данных.

* Переход в директорию src/Model/Table
  + Cd ..
  + cd src/Model/Table
* Создать файл UsersTable.php
* Выбрать таблицу:

<?php

namespace App\Model\Table;

use Cake\ORM\Table;

class UsersTable extends Table {

public function initialize(array $config): void {

$this->setTable('users');

}}

Загрузка на хостинг.

Перейти по предоставленной гиперссылке на платформу Beget: https://beget.com/ru/hosting/free.

Нажать на интерфейсный элемент «Сайты» для отображения URL-адреса созданного веб-ресурса. Переместиться на целевой веб-ресурс и визуально подтвердить наличие стандартной приветственной страницы Beget.

Вернуться в раздел управления БД, скопировать параметры подключения (наименование БД, логин, пароль, хост) и модифицировать соответствующие параметры в файле connect.php.

Для передачи объемных файлов на сервер использовать один из методов:

* FTP-подключение (не рекомендуется из-за потенциальных ошибок, вызванных работой прокси-серверов);
* Загрузка ZIP-архива через файловый менеджер.
* Запаковать целевую директорию в архив формата ZIP. Через панель управления Beget открыть файловый менеджер.
* Переместить в директорию public\_html, связанную с доменом проекта. Удалить все существующие файлы. Загрузить подготовленный архив через интерфейс загрузки, после чего распаковать его содержимое средствами файлового менеджера. Удалить исходный архив.

Перейти на URL веб-ресурса для проверки корректности развертывания. При наличии ошибок — провести анализ системных сообщений и внести необходимые правки. Учесть, что прямое редактирование минифицированных файлов на сервере недоступно. Рекомендуется очистить кэш браузера при некорректном отображении контента.

# Список использованной литературы

1. Паршукова Н. Б. Программирование с использованием PHP и MySQL в разработке веб-приложений. – 2021.
2. Байдыбеков А. А., Гильванов Р. Г., Молодкин И. А. Современные фреймворки для разработки web-приложений //Интеллектуальные технологии на транспорте. – 2020. – №. 4 (24). – С. 23-29.
3. Кряжева Е. В., Васина Т. А. Общие подходы к проектированию ВЕБ-приложений //Заметки ученого. – 2021. – №. 9-2. – С. 32-36.
4. Биганяков А. А. Архитектура современных веб-приложений: паттерн проектирования MVC //Конкурентоспособность территорий. – 2019. – С. 141-143.
5. Прохоренок Н., Дронов В. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера, 5 изд. – БХВ-Петербург, 2019.
6. Никсон Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5. 6-е изд. – Питер, 2023.
7. 2
8. <https://book.cakephp.org/2/_downloads/en/CakePHPCookbook.pdf>