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 Yii2 В РАЗРАБОТКЕ ВЕБ-ПРИЛОЖЕНИЯ «Социальная сеть»

Разработчик: студент группы 3ИСПР2

Морозов М.Р.

(подпись)

Оценка защиты курсового проекта

« »

Дата защиты

« » 2024

Руководитель проекта:

преподаватель Степанов С.О.

(подпись)

Ногинск, 2024 г

Титульный лист

**Введение**

**Цель:** В моем курсовом проекте я должен был создать социальную сеть. Я её создал с помощью таких инструментов, как Bootstrap и Yii2 с нужным функционалом.

**Актуальность:** Социальные сети стали неотъемлемой частью нашей жизни. Сегодня они предоставляют уникальные возможности связи, обмена информацией и взаимодействия со всем миром. Давайте рассмотрим роль и влияние социальных сетей в современном мире. В первую очередь, социальные сети играют важную роль в поддержании связей между друзьями и родственниками. Вместо того чтобы тратить много времени и сил на организацию встреч и звонки, мы можем легко и быстро обмениваться новостями, фотографиями и видео через социальные сети. Это позволяет нам чувствовать себя ближе к людям, которых мы не видели уже долгое время, а также легко поддерживать коммуникацию с теми, кто находится на другом конце света.

Между тем, социальные сети не только усиливают нашу коммуникацию и самовыражение, но и оказывают значительное влияние на наш потребительский опыт и общественные процессы. Сегодня многие компании используют социальные сети для продвижения своих продуктов и услуг, благодаря чему потребители могут быстро получать информацию о новинках, акциях и специальных предложениях. Кроме того, социальные сети помогают людям обмениваться отзывами и рекомендациями, что делает процесс выбора товаров и услуг более информированным и удобным.

Однако, социальные сети имеют и свои темные стороны. Все больше и больше людей страдают от зависимости от социальных сетей, проводя в них часы вместо того, чтобы находиться в реальном мире и уделять внимание реальным отношениям. Кроме того, социальные сети стали площадкой для распространения ложных новостей, манипуляции и ненавистного поведения. Безответственное использование социальных сетей может привести к серьезным последствиям для ментального и эмоционального здоровья людей, а также для общественной безопасности в целом. Влияние социальных сетей на ИТ-рынок огромно, и все больше и больше людей используют социальные сети, чтобы быть в курсе того, что происходит в мире технологий. С появлением социальных сетей, таких как Вконтакте, Яндекс и Google+, компании используют эти платформы для того, чтобы максимально увеличить свое присутствие и охват.

Социальные сети демонстрируют свое разнообразие и стабильный рост пользователей и прибыли, предлагая клиентам различные функции для общения с близкими людьми. Для долгой и успешной работы в этой сфере важно следить за актуальными новостями, улучшать свою социальную сеть и подстраиваться к новым потребностям пользователя.

**Задачи:**

1. Выбрать стиль к курсовому проекту.
2. Изучить методический форум для Yii2.
3. Изучить Bootstrap.
4. Сделать социальную сеть при помощи этих инструментов.
5. Загрузка социальной сети на выбранный хостинг сервер.
6. Написание и защита курсового проекта.

**Основная часть**

**1.Общее введение о разработке веб-приложения на фреймворке Yii2.**

В мире ИТ существует множество инструментов, технологий, фреймворков, которые в свою очередь помогают будущим ИТ-персоналу создавать и загружать свои проекты во всемирную паутину интернет.

Yii – это универсальный фреймворк, и может быть задействован во всех типах веб-приложений. Если вы уже знакомы с другими фреймворками, вам наверняка будет интересно сравнить их с Yii:

1. Как и многие другие PHP фреймворки, для организации кода Yii использует архитектурный паттерн MVC (Model-View-Controller).
2. Yii придерживается философии простого и элегантного кода, не пытаясь усложнять дизайн только ради следования каким-либо шаблонам проектирования.
3. Yii является полным набором фреймворков и включает в себя проверенные и хорошо зарекомендовавшие себя возможности, такие как ActiveRecord для реляционных и NoSQL баз данных, поддержку REST API, многоуровневое кэширование и другие.
4. Yii отлично расширяем. Вы можете настроить или заменить практически любую часть основного кода. Используя архитектуру расширений, легко делиться кодом или использовать код сообщества.
5. Одна из главных целей Yii – производительность.

Yii — не проект одного человека. Он поддерживается и развивается сильной командой и большим сообществом разработчиков, которые ей помогают. Авторы фреймворка следят за тенденциями веб-разработки и развитием других проектов. Наиболее подходящие возможности и лучшие практики регулярно внедряются в фреймворк в виде простых и элегантных интерфейсов. Благодаря его компонентной структуре и отличной поддержке кэширования, фреймворк особенно подходит для разработки таких крупных проектов, как порталы, форумы, CMS, магазины или RESTful-приложения. Yii включает в себя удобный инструмент Gii, который обеспечивает быструю среду выполнения путем создания часто используемых фрагментов кода, а также полных CRUD-контроллеров. Gii — инструмент для быстрой генерации кода, который мне очень понравился, и я буду использовать его в своих будущих проектах. Чтобы следовать этому руководству, вам потребуется базовое понимание PHP. При установке Yii автоматически будет установлено расширение Gii.

На данный момент существует две основные ветки Yii: 1.1 и 2.0. Ветка 1.1 является предыдущим поколением и находится в состоянии поддержки. Версия 2.0 – это полностью переписанный Yii, использующий последние технологии и протоколы, такие как Composer, PSR, пространства имен, трейты и многое другое. 2.0 — текущее поколение фреймворка. На этой версии будут сосредоточены основные усилия несколько следующих лет. Для работы Yii 2.0 необходим PHP 5.4 или выше. Данная версия включает большое количество улучшений по сравнению с версией 5.2, которая использовалась Yii 1.1. Таким образом, существует много различий в языке, которые вы должны принимать во внимание:

* Пространства имён;
* Анонимные функции;
* Использование короткого синтаксиса для массивов: [...элементы...] вместо array(...элементы...);
* Использование короткого echo <?= для вывода в файлах представлений. С версии PHP 5.4 данную возможность можно использовать не опасаясь;
* Классы и интерфейсы SPL;
* Позднее статическое связывание (LSB);
* Классы для дат и времени;
* Трейты;

Интернационализация (intl); Yii 2.0 использует расширение PHP intl для различного функционала интернационализации.

**2.Загрузка всех необходимых программ и фреймворка для работы с курсовым проектом.**

Для работы с Yii2 мне потребовалось скачать несколько необходимых программ, а именно Phpstorm 2024 1.3, Composer, OSPanel, Git.

Для установки Git нам просто нужно перейти на сайт GitHub, зарегистрироваться на их официальном сайте и потом скачать Git на свой компьютер. Git — это специальная программа, которая позволяет отслеживать любые изменения в файлах, хранить их версии и оперативно возвращаться в любое сохранённое состояние.



Рис.1.установка Git

Для установки PhpStorm нужно зарегистрироваться на платформе Jetbrains IDE, после чего скачать пробную бесплатную версию PhpStorm на их официальном сайте, на свой компьютер. PhpStorm нет равных по функциональности редактора, который учитывает все особенности языка. Умное автодополнение кода и рефакторинги, мгновенное обнаружение ошибок, вывод типов, быстрая и удобная навигация по кодовой базе и множество других функций избавляют от скучных и однообразных задач при написании кода.

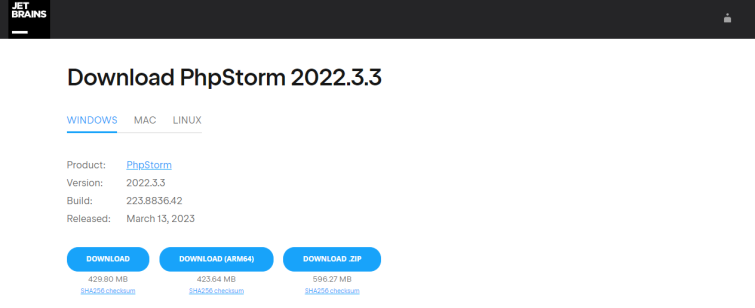


Рис.2.установка PhpStorm

Для установки OSPanel, нам нужно зайти на их официальный сайт, выбрать и скачать нужную нам версию OSPanel. Open Server Panel — это портативный программный комплекс, созданный для того чтобы помочь веб-мастерам в разработке, отладке и тестировании сайтов непосредственно на компьютере (даже если на нем нет интернета) под управлением ОС Windows.

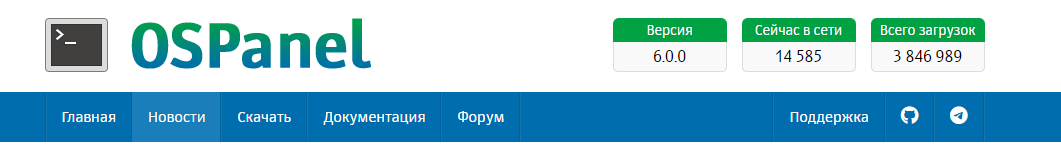


Рис.4. установка OSPanel

После того, как мы установили Open Server (OSPanel) нам нужно скачать Composer, для этого нам нужно тоже перейти на их официальный сайт и скачать exe файл.

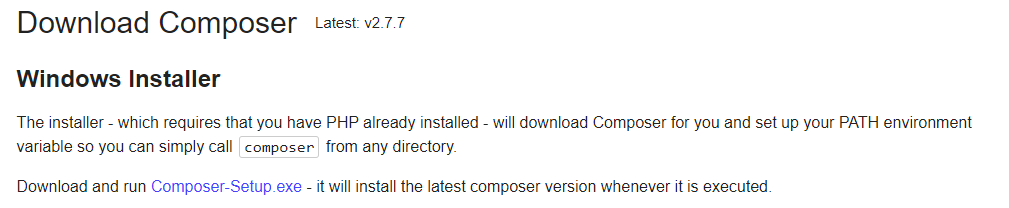


Рис.5.установка Composer

После установки всех программ, я создал репозиторий курсовая Морозов, после чего в Phpstorm, я открыл терминал и написал composer create-project –prefer-dist yiisoft/Kursovay-Morozov basic, тем самым, эта команда позволила мне установить последнюю версию Yii в директорию Kursovay-Morozov, после установки нужных мне файлов, в терминале я прописал php yii serve, тем самым я запустил отдельный сервер и проверил, как работает Yii.

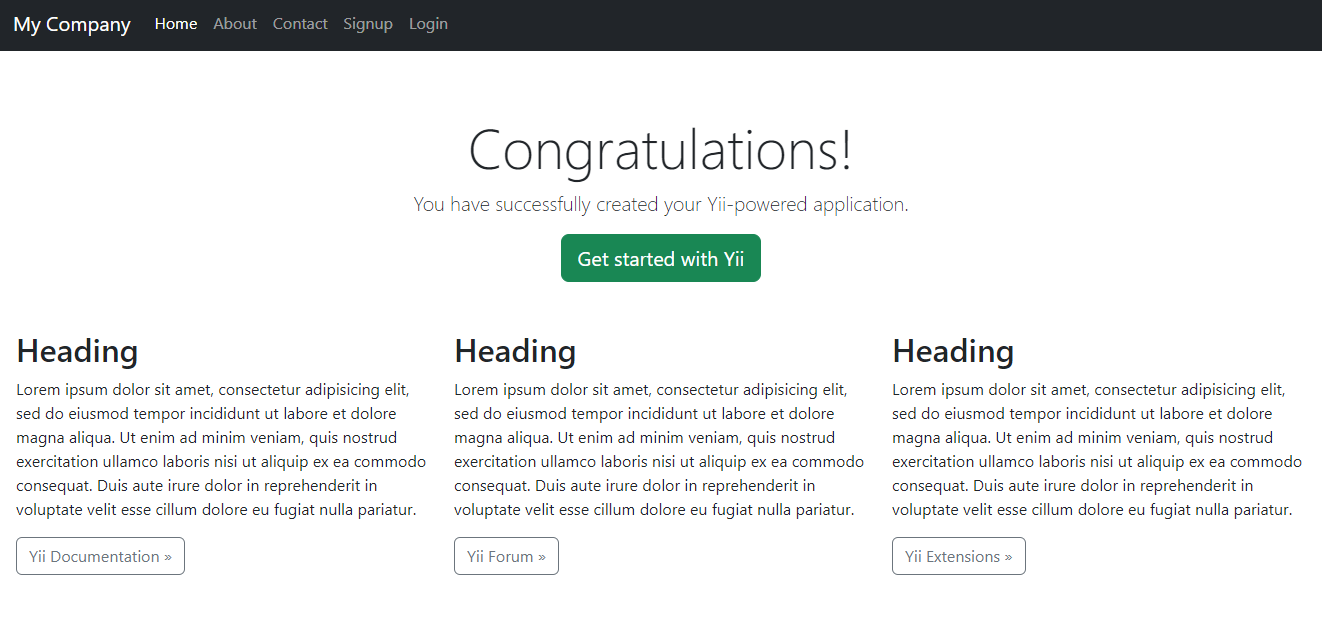


Рис.6 установка Yii

**3.Разработка социальной сети.**

**3.1 Backend часть(Php).**

В разработке социальной сети, я решил начать работу с Backend, чтобы посмотреть все функционалы, которые мне надо сделать. Для начала мне нужно было создать регистрацию пользователя, по моему логическому мнению - это самое легкое, что можно сделать. Сначала я создал миграцию (миграция - создание базы данных, где будет храниться вся информация о пользователе: его пароль, почта и никнейм). Для этого в Phpstorm, я нажал на кнопку терминал, и написал следующую строчку кода: yii migrate/create user table. Тем самым у меня создалась миграция, где я прописал следущую команду:

public function safeUp()

{

$this->createTable('user', [

'id' => $this->primaryKey(11)->unsigned(),

'email' => $this->string(64),

'username' => $this->string(32),

'password\_hash' => $this->string(60),

'salt' => $this->string(32),

'auth\_key' => $this->string(32),

'verification\_token' => $this->string(32),

'status' => $this->tinyInteger(1)->unsigned()->notNull()->defaultValue(2),

]);

}

public function safeDown()

{

$this->dropTable('user');

}

}

$this->createTable('user', [...] ); - Этот метод создает новую таблицу в базе данных с именем user.

Второй аргумент метода createTable представляет собой массив определений столбцов таблицы. В данном случае определены следующие столбцы:

* 'id' => $this->primaryKey(11)->unsigned() - Этот столбец id будет первичным ключом таблицы (primaryKey(11) указывает тип ключа и его размер, а unsigned() указывает, что значение не может быть отрицательным).
* 'email' => $this->string(64) - Столбец для хранения адреса электронной почты, тип VARCHAR(64) (максимальная длина 64 символа).
* 'username' => $this->string(32) - Столбец для хранения имени пользователя, тип VARCHAR(32) (максимальная длина 32 символа).
* 'password\_hash' => $this->string(60) - Столбец для хранения хеша пароля, тип VARCHAR(60) (максимальная длина 60 символов).
* 'salt' => $this->string(32) - Столбец для хранения соли, используемой при хешировании пароля, тип VARCHAR(32) (максимальная длина 32 символа).
* 'auth\_key' => $this->string(32) - Столбец для хранения ключа аутентификации (auth key), тип VARCHAR(32) (максимальная длина 32 символа).
* 'verification\_token' => $this->string(32) - Столбец для хранения токена верификации, тип VARCHAR(32) (максимальная длина 32 символа).
* 'status' => $this->tinyInteger(1)->unsigned()->notNull()->defaultValue(2) - Столбец для хранения статуса пользователя. Тип TINYINT(1) (число от 0 до 255). Указаны следующие параметры:
  + unsigned() - значение не может быть отрицательным.
  + notNull() - столбец не может содержать NULL.
  + defaultValue(2) - значение по умолчанию равно 2.

Эта миграция создает таблицу user с указанными столбцами и их настройками. Каждый столбец представляет различные атрибуты пользователя, такие как электронная почта, имя пользователя, хеш пароля, соль, ключ аутентификации, токен верификации и статус.

После создания базы данных я решил создать контроллер регистрации, где код будет сохранять и проверять пароли и никнейм пользователя.

return [

'access' => [

'class' => AccessControl::className(),

'only' => ['logout', 'login', 'signup'],

'rules' => [

[

'actions' => ['logout'],

'allow' => true,

'roles' => ['@'],

],

[

'allow' => true,

'actions' => ['login', 'signup'],

'roles' => ['?'],

],

],

],

**Поведение access:**

* 'class' => AccessControl::className() указывает, что используется класс AccessControl, который обеспечивает управление доступом.
* 'only' => ['logout', 'login', 'signup'] указывает, что управление доступом применяется только к действиям logout, login и signup.
* 'rules' => [...] определяет правила доступа для каждого действия.
*  'actions' => ['logout'] указывает, что это правило применяется только к действию logout.
* 'allow' => true разрешает доступ, если выполняются следующие условия:

Пользователь авторизован ('@' означает, что требуется авторизованный пользователь).

 'actions' => ['login', 'signup'] указывает, что это правило применяется к действиям login и signup.

 'allow' => true разрешает доступ, если выполняются следующие условия:

Пользователь не авторизован ('?' означает, что требуется гость).

**Поведение verbs:**

* 'class' => VerbFilter::className() указывает, что используется класс VerbFilter, который фильтрует запросы по HTTP-методам (например, GET, POST, PUT, DELETE).
* 'actions' => [...] определяет разрешенные HTTP-методы для каждого действия.

Таким образом, метод behaviors() в контроллере SiteController определяет правила доступа (AccessControl) и фильтр HTTP-методов (VerbFilter), обеспечивая контроль над тем, какие действия могут быть выполнены пользователем в зависимости от его авторизации и HTTP-метода запроса. После чего создал несколько моделей, чтобы они представляли собой объекты бизнес данных, правил и логики. Первая модель это SighupForm.php:

public $username;

public $email;

public $password;

public function rules()

{

return [

[['username', 'email', 'password'], 'required'],

];

}

**Публичные свойства:** public $username, public $email, public $password - это свойства класса, которые соответствуют полям формы регистрации: имя пользователя, электронная почта и пароль.

Метод rules()Определяет правила валидации данных, которые будут применяться при попытке сохранения модели.[['username', 'email', 'password'], 'required'] указывает, что поля username, email и password обязательны для заполнения при валидации.Метод signup()Отвечает за процесс регистрации нового пользователя.if ($this->validate()) проверяет валидность введенных данных на основе правил валидации из метода rules().getUser() вызывается для создания экземпляра модели User, заполнения его данными из формы (username, email, password) и сохранения в базу данных.Если сохранение проходит успешно ($user->save() возвращает true), выводится сообщение об успешной регистрации через Yii::$app->session->setFlash('success', ...), и метод возвращает true.

Если данные не прошли валидацию или сохранение не удалось, метод возвращает false. Метод getUser() Создает новый экземпляр модели User. Заполняет свойства username, email и устанавливает пароль с помощью метода setPassword() из модели User. Возвращает экземпляр модели User, готовый для сохранения в базу данных. Потом я создал модель LoginForm.php, где я прописал следующую строчку кода:

{

public $username;

public $password;

public $rememberMe = true;

public function rules()

{

return [

// username and password are both required

['username', 'validateStatus'],

[['username', 'password'], 'required'],

// rememberMe must be a boolean value

['rememberMe', 'boolean'],

// password is validated by validatePassword()

['password', 'validatePassword']

];

}

Класс LoginForm выполняет следующие основные функции:

* Проверка введенных данных пользователя (имя пользователя, пароль).
* Проверка статуса пользователя (например, заблокирован ли аккаунт).
* Вход пользователя в систему при успешной валидации.

Этот класс является важной частью механизма аутентификации и авторизации веб-приложения на Yii2, обеспечивая безопасный и контролируемый доступ пользователей к системе.

**Публичные свойства:**

public $username, public $password, public $rememberMe - это свойства класса, которые соответствуют полям формы входа: имя пользователя, пароль и флаг "запомнить меня". Метод rules()Определяет правила валидации данных, которые будут применяться при попытке входа [['username', 'password'], 'required'] указывает, что поля username и password обязательны для заполнения при валидации.['rememberMe', 'boolean'] проверяет, что свойство rememberMe является логическим значением (true или false)['password', 'validatePassword'] указывает, что для валидации пароля будет использоваться метод validatePassword().

Метод validatePassword($attribute) Проверяет корректность введенного пароля. Если нет других ошибок валидации (!$this->hasErrors()), получает пользователя с помощью метода getUser(). Если пользователь не найден (!$user) или пароль не проходит проверку с помощью метода validatePassword() у объекта пользователя, добавляет ошибку валидации для атрибута $attribute с сообщением 'Incorrect username or password.'.

Метод getUser() Ищет пользователя по его имени (username) с помощью статического метода findByUsername() модели User. Возвращает объект пользователя или null, если пользователь не найден.

Метод validateStatus()Проверяет статус пользователя после получения его с помощью метода getUser(). Если статус пользователя равен User::STATUS\_BLOCKED (заблокирован), добавляет ошибку валидации для атрибута 'status' и устанавливает flash сообщение об ошибке для текущей сессии.

Метод login() Вызывается при попытке входа пользователя в систему. Если данные формы проходят валидацию ($this->validate()), вызывает Yii::$app->user->login() для входа пользователя. При успешном входе возвращает true, иначе false.

После этого я создал еще одну модель я регистрации это User.php.:

class User extends ActiveRecord implements IdentityInterface

{

const STATUS\_BLOCKED = 0;

const STATUS\_ACTIVE = 1;

public $role;

**Константы и методы**: const STATUS\_BLOCKED и const STATUS\_ACTIVE определяют статусы пользователя. Методы tableName(), rules(), attributeLabels() определяют соответственно таблицу базы данных, правила валидации и метки атрибутов.

**Методы для работы с идентификацией и авторизацией**: findIdentity($id) и findIdentityByAccessToken($token, $type) позволяют Yii2 искать пользователя по идентификатору или токену доступа. validatePassword($password) и setPassword($password) для проверки и установки пароля с использованием хэширования.

**Методы для работы со статусами и ролями**: getStatuses() возвращает массив статусов. getRoles() и getRolesArray() возвращают роли пользователя. Методы для работы с профилями и связями: getProfile(), getFollowers(), getFriendships(), getConversations() и другие методы предоставляют доступ к профилю пользователя, его подписчикам, друзьям и сообщениям. Методы для работы с токенами и проверкой: generateAuthKey() и generateVerificationToken() генерируют аутентификационный ключ и токен подтверждения. sendConfirmEmail() и verify() используются для отправки подтверждения по электронной почте и подтверждения пользователя. Методы для обработки сохранения и валидации: beforeSave() используется для подготовки модели перед сохранением в базу данных. save() сохраняет модель в базу данных, учитывая также связанные сущности (например, профиль пользователя и его роль). Этот код представляет собой полнофункциональную модель пользователя (User) для веб-приложения, которая обеспечивает управление пользователями, их аутентификацию, авторизацию, управление ролями и другие связанные с этим операции.

После создания регистрации я решил создать профиль. Для этого я создал новую миграцию и назвал её profile\_table где прописан код:

public function safeUp()

{

$this->createTable('profile', [

'id' => $this->primaryKey(11)->unsigned(),

'avatar' => $this->string(64),

'first\_name' => $this->string(32),

'last\_name' => $this->string(32),

'birth\_date' => $this->string(32),

'details' => $this->text(),

'user\_id' => $this->integer(11)->unsigned(),

'gallery\_id' => $this->integer(11)->unsigned(),

]);

**Метод safeUp()**:

* Создает таблицу profile с колонками:
  + id: первичный ключ (integer, 11 знаков, без знака)
  + avatar: строка до 64 символов для хранения ссылки на аватар
  + first\_name и last\_name: строки до 32 символов для имени и фамилии
  + birth\_date: строка до 32 символов для даты рождения
  + details: текстовое поле для дополнительных данных
  + user\_id и gallery\_id: связи с таблицами user и gallery через внешние ключи
* Создает индексы и внешние ключи для столбцов user\_id и gallery\_id, связывая их с таблицами user и gallery.

**Метод safeDown()**:

* Откатывает миграцию:
  + Удаляет внешние ключи и индексы для столбцов user\_id и gallery\_id.
  + Удаляет таблицу profile.

Эта миграция обеспечивает безопасное создание и удаление таблицы profile с необходимыми связями с другими таблицами в базе данных Yii2-приложения.

После создания миграции я создал контроллер с название UserController c кодом:

public function actionDisplay($id)

{

$model = $this->findModel($id);

$dataProvider = new ActiveDataProvider([

'query' => $model->getFollowers(),

'pagination' => [

'pageSize' => 3

]

]);

* Начало формы
* Конец формы

Этот код представляет собой конфигурацию поведений (behaviors) в Yii Framework. В данном случае, он определяет правила доступа к действиям контроллера в зависимости от роли пользователя. Пользователи с ролью admin имеют доступ ко всем действиям без ограничений. Пользователи с ролью moderator имеют доступ к действиям: update, index, display, followers, find, follow, unfollow, reset-image, view. Пользователи с ролью user имеют доступ к действиям: display, followers, find, follow, unfollow, reset-image. Дополнительно, для действия update, разрешено выполнение только если у пользователя есть разрешение updateOwnProfile и если user\_id в запросе соответствует user\_id текущего пользователя.

Этот код обеспечивает управление доступом к различным функциям в зависимости от роли пользователя и специфических разрешений для определенных действий. После создание контроллера я создал модель Profile.php где написан след код:

public function attributeLabels()

{

return [

'id' => 'ID',

'first\_name' => 'First Name',

'last\_name' => 'Last Name',

'birth\_date' => 'Birthday',

'details' => 'Details',

'user\_id' => 'User ID',

'gallery\_id' => 'Gallery ID',

];

}

Этот код обеспечивает выполнения редактирования профиля любого пользователя, который зарегистрировался на сайт:

['avatar'], 'string – эта строчка позволяет отображать фотографию пользователя в его профиле.

[['imageFile'], 'image', 'extensions' => ['png', 'jpg']] -эта строчка позволяет пользователю загрузить любую фотографию, тем самым она будет отображаться в его профиле.

[['first\_name', 'last\_name', 'birth\_date'], 'string', 'max' => 32],

[['first\_name', 'last\_name', 'birth\_date', 'details'], 'default', 'value' => null],- эти строчки кода позволяют пользователю вводить в базу данных имя и фамилию, дату его рождения и небольшую информацию о пользователе (статут).

public function getGallery()  
{  
 return $this->hasOne(Gallery::*className*(), ['id' => 'gallery\_id']);  
}  
  
*/\*\*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/*public function getUser()  
{  
 return $this->hasOne(User::*className*(), ['id' => 'user\_id']);  
}  
  
public function getName()  
{  
 return isset($this->first\_name) ? $this->getFullName() : $this->getUsername();  
}  
  
*/\*\*  
 \** ***@return*** *string  
 \*/*public function getFullName()  
{  
 return $this->first\_name . ' ' . $this->last\_name;  
}  
  
*/\*\*  
 \** ***@return*** *string  
 \*/*public function getUsername()  
{  
 return $this->user->username;  
}  
  
public function beforeSave($insert)  
{  
 if (parent::*beforeSave*($insert)) {  
 if ($imageFile = UploadedFile::*getInstance*($this, 'imageFile')) {  
 $this->avatar = '/uploads/' . $imageFile->baseName . '.' . $imageFile->extension;  
 $imageFile->saveAs(substr($this->avatar, 1));  
 }  
 return true;  
 }  
 return false;  
}  
  
public function setDefaultImage()  
{  
 $this->avatar = '/gallery/standart.png';  
}

рис.7. небольшие правила работы модели

После создания профилей, я решил создать добавления в друзья. Для этого мне понадобилась одни миграция и одна модель, потому что в UserController у меня было прописано, как 1 пользователь должен добавить другого. В миграции я создал такой код:

public function safeUp()  
{  
 $this->createTable('friendship', [  
 'id' => $this->primaryKey(11)->unsigned(),  
 'first\_user\_id' => $this->integer(11)->unsigned(),  
 'second\_user\_id' => $this->integer(11)->unsigned(),  
 ]);  
  
 $this->createIndex(  
 'idx-friendship-users\_id',  
 'friendship',  
 [  
 'first\_user\_id',  
 'second\_user\_id'  
 ],  
 true  
 );  
  
 // add foreign key for table `user`  
 $this->addForeignKey(  
 'fk-friendship-first\_user\_id',  
 'friendship',  
 'first\_user\_id',  
 'user',  
 'id',  
 'CASCADE',  
 'CASCADE'  
 );  
  
 // add foreign key for table `user`  
 $this->addForeignKey(  
 'fk-friendship-second\_user\_id',  
 'friendship',  
 'second\_user\_id',  
 'user',  
 'id',  
 'CASCADE',  
 'CASCADE'  
 );  
}

Рис.8 показ миграции добавления в друзья

Предоставленный вами код представляет собой файл миграции, написанный на PHP для создания таблицы с именем «дружба» со столбцами «id» (первичный ключ), «first\_user\_id» и «second\_user\_id». Он также включает определение индекса для столбцов first\_user\_id и Second\_user\_id для обеспечения уникальности пар идентификаторов пользователей в таблице дружбы.

Кроме того, в столбец «first\_user\_id» добавляется ограничение внешнего ключа, ссылающееся на столбец «id» таблицы «user», с каскадным выполнением действий обновления и удаления.

В целом, эта миграция устанавливает схему базы данных для управления дружбой между пользователями и обеспечивает целостность данных посредством отношений внешнего ключа.

После чего я создал модель под название Friendship.php с исходным кодом:

public function rules()  
{  
 return [  
 [['first\_user\_id', 'second\_user\_id'], 'integer'],  
 [['first\_user\_id', 'second\_user\_id'], 'unique', 'targetAttribute' => ['first\_user\_id', 'second\_user\_id']],  
 [['first\_user\_id'], 'exist', 'skipOnError' => true, 'targetClass' => User::*className*(), 'targetAttribute' => ['first\_user\_id' => 'id']],  
 [['second\_user\_id'], 'exist', 'skipOnError' => true, 'targetClass' => User::*className*(), 'targetAttribute' => ['second\_user\_id' => 'id']],  
 ];  
}  
  
*/\*\*  
 \* {****@inheritdoc****}  
 \*/*public function attributeLabels()  
{  
 return [  
 'id' => 'ID',  
 'first\_user\_id' => 'First User ID',  
 'second\_user\_id' => 'Second User ID',  
 ];  
}

Рис. 9 показ модели добавления друзья

**Сопоставление структуры таблицы** : Класс Friendship соответствует таблице «дружба» в базе данных. Он имеет атрибуты ( id , first\_user\_id , second\_user\_id ), которые отражают столбцы в таблице базы данных. Класс Friendship определяет отношения с User классом с помощью hasOne метода. Эти отношения представляют собой связь между объектами Friendship и User на основе столбцов «first\_user\_id» и «second\_user\_id».

**Правила проверки :** Метод rules() определяет правила проверки атрибутов модели. Он обеспечивает это first\_user\_id и second\_user\_id должен быть целым числом, а комбинация этих двух столбцов должна быть уникальной в таблице «дружба». Это также гарантирует, что значения first\_user\_id и second\_user\_id должны существовать в столбце «id» таблицы User .

Этот attributeLabels() метод предоставляет удобочитаемые метки атрибутов для целей отображения.

Таким образом, этот код определяет класс модели Yii2 для таблицы «дружбы» с сопоставлениями атрибутов, отношениями с моделью User , правилами проверки и метками атрибутов для атрибутов. Это позволяет приложению взаимодействовать и манипулировать данными о дружбе в базе данных.

После выполненной работы, мне осталось только сделать, только личный чат, что я и начал делать. Я создал миграцию chat\_table:

public function safeUp()  
{  
 $this->createTable('conversation', [  
 'id' => $this->primaryKey(11)->unsigned(),  
 'title' => $this->string(64),  
 ]);  
  
 $this->createTable('user\_has\_conversation', [  
 'id' => $this->primaryKey(11)->unsigned(),  
 'user\_id' => $this->integer(11)->unsigned(),  
 'conversation\_id' => $this->integer(11)->unsigned()  
 ]);

Рис.10 Демонстрация миграции

**Создание таблиц :** Метод safeUp() создает следующие три таблицы:

* conversation : Содержит столбцы id (первичный ключ) и title .
* user\_has\_conversation : Содержит столбцы id (первичный ключ), user\_id и conversation\_id .
* conversation\_message : Содержит столбцы id (первичный ключ), conversation\_id , user\_id , content и created\_at .

**Создание индексов** : Он создает индексы для таблиц user\_has\_conversation и conversation\_message для повышения производительности запросов.

**Добавление внешних ключей :** Он устанавливает ограничения внешнего ключа между таблицами:

* conversation\_message.user\_id Рекомендации user.id .
* conversation\_message.conversation\_id Рекомендации conversation.id .
* user\_has\_conversation.user\_id Рекомендации user.id .
* user\_has\_conversation.conversation\_id Рекомендации conversation.id .

**Удаление таблиц** : Метод safeDown() удаляет таблицы в порядке, обратном их созданию. Он также удаляет добавленные индексы и ограничения внешнего ключа.

Таким образом, этот класс миграции помогает настроить необходимую схему базы данных для системы чата путем создания таблиц, индексов и установления связей с использованием внешних ключей. Он также предоставляет возможность откатить изменения, удалив таблицы и связанные объекты базы данных.

Потом у меня идет создание контроллера с названием СonversationController.php, в котором содержится следующий код:

public function behaviors()  
{  
 return [  
 'access' => [  
 'class' => AccessControl::*className*(),  
 'only' => ['logout', 'login', 'signup'],  
 'rules' => [  
 [  
 'actions' => ['logout'],  
 'allow' => true,  
 'roles' => ['@'],  
 ],  
 [  
 'allow' => true,  
 'actions' => ['login', 'signup'],  
 'roles' => ['?'],  
 ],  
 ],  
 ],  
 'verbs' => [  
 'class' => VerbFilter::*className*(),  
 'actions' => [  
 'logout' => ['post'],  
 ],  
 ],  
 ];  
}public function actions()  
{  
 return [  
 'error' => [  
 'class' => 'yii\web\ErrorAction',  
 ],  
 'captcha' => [  
 'class' => 'yii\captcha\CaptchaAction',  
 'fixedVerifyCode' => *YII\_ENV\_TEST* ? 'testme' : null,  
 ],  
 ];  
}  
  
*string  
 \*/*public function actionIndex()  
{  
 return $this->render('index');  
}  
  
public function actionLogin()  
{  
 if (!Yii::*$app*->user->isGuest) {  
 return $this->goHome();  
 }  
  
 $model = new LoginForm();  
 if ($model->load(Yii::*$app*->request->post()) && $model->login()) {  
 return $this->goBack();  
 }  
  
 $model->password = '';  
 return $this->render('login', [  
 'model' => $model,  
 ]);  
}

Рис.11 Демонстрация контроллера чата.

* Index: Выводит список всех разговоров вместе с их соответствующими участниками.
* View: Отображает детали указанного разговора.
* GetConversation: Создает новый разговор или извлекает существующий диалог между двумя пользователями.
* Display: Отображает текущий разговор, позволяет пользователям отправлять сообщения и отображает все сообщения в базу данных.

После контроллера я создал модель conversation\_message, где показан следующий код:

<?php  
  
namespace app\models;  
  
use Yii;  
use yii\behaviors\TimestampBehavior;  
use yii\db\ActiveRecord;  
  
*/\*\*  
 \* This is the model class for table "conversation\_message".  
 \*  
 \** ***@property*** *string $id  
 \** ***@property*** *string $conversation\_id  
 \** ***@property*** *string $user\_id  
 \** ***@property*** *string $content  
 \** ***@property*** *string $created\_at  
 \*  
 \** ***@property*** *Conversation $conversation  
 \** ***@property*** *User $user  
 \*/*class ConversationMessage extends ActiveRecord  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public static function tableName()  
 {  
 return 'conversation\_message';  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function rules()  
 {  
 return [  
 [['conversation\_id', 'user\_id'], 'integer'],  
 [['content'], 'string'],  
 [['created\_at'], 'string', 'max' => 32],  
 [['conversation\_id'], 'exist', 'skipOnError' => true, 'targetClass' => Conversation::*className*(), 'targetAttribute' => ['conversation\_id' => 'id']],  
 [['user\_id'], 'exist', 'skipOnError' => true, 'targetClass' => User::*className*(), 'targetAttribute' => ['user\_id' => 'id']],  
 ];  
 }  
  
 public function behaviors()  
 {  
 return [  
 'timestamp' => [  
 'class' => TimestampBehavior::*className*(),  
 'attributes' => [  
 ActiveRecord::*EVENT\_BEFORE\_INSERT* => ['created\_at'],  
 ],  
 'value' => date('Y-m-d H:i:s'),  
 ],  
 ];  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function attributeLabels()  
 {  
 return [  
 'id' => 'ID',  
 'conversation\_id' => 'Conversation ID',  
 'user\_id' => 'User ID',  
 'content' => 'Content',  
 'created\_at' => 'Created At',  
 ];  
 }  
  
 */\*\*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* public function getConversation()  
 {  
 return $this->hasOne(Conversation::*className*(), ['id' => 'conversation\_id']);  
 }  
  
 public function getAuthor()  
 {  
 return $this->user->getUsername();  
 }  
  
 */\*\*  
 \** ***@param*** *User $user  
 \** ***@return*** *bool  
 \*/* public function isOwner($user)  
 {  
 if ($user === $this->getUser()) {  
 return true;  
 }  
 if ($user === null) {  
 return false;  
 }  
 if ($this->getUser() === null) {  
 return false;  
 }  
 return $user->getId() === $this->user->getId();  
 }  
  
 */\*\*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* public function getUser()  
 {  
 return $this->hasOne(User::*className*(), ['id' => 'user\_id']);  
 }  
}

Рис.12 Демонстрация модели чата

Он содержит описание столбцов таблицы и их свойств, а также связи с другими моделями данных. Каждая строка таблицы "conversation\_message" представлена объектом этого класса. Он содержит свойства, такие как id, conversation\_id, user\_id, content и created\_at, которые соответствуют столбцам таблицы. Также в этом классе определены правила валидации данных для каждого свойства, а также методы для получения связанных объектов Conversation и User. Другие методы в этом классе позволяют проверить, является ли пользователь владельцем сообщения и получить имя автора сообщения. Также в этом классе определено поведение "timestamp", которое автоматически устанавливает значение "created\_at" при создании новой записи. В целом, этот код определяет структуру данных и их взаимосвязи для таблицы "conversation\_message" и предоставляет методы для работы с этими данными.

**3.2Frontend часть(Bootstrap).**

После написания функций и создания таблиц, я решил заняться Frontend, сделать функционалу более понятный интерфейс. Для этого в PhpStorm я открыл папку view, и начал добавлять, туда весь графический интерфейс. Первым, что я затронул это главную страницу.

<?php $this->beginPage() ?>  
<!DOCTYPE html>  
<html lang="<?= Yii::*$app*->language ?>">  
<head>  
 <meta charset="<?= Yii::*$app*->charset ?>">  
 <meta http-equiv="X-UA-Compatible" content="IE=edge">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <?= Html::*csrfMetaTags*() ?>  
 <title><?= Html::*encode*($this->title) ?></title>  
 <?php $this->head() ?>  
</head>  
<body>  
<?php $this->beginBody() ?>  
  
<div class="wrap">

Рис.13 Показ главной страницы.

Он содержит разметку для отображения основных элементов веб-страницы, таких как навигационное меню, хлебные крошки (breadcrumbs), сообщения об ошибках (Alert), а также область контента. Код также определяет, какие элементы будут отображаться в навигационном меню в зависимости от статуса пользователя и их роли. Например, для авторизованных пользователей отображаются ссылки на страницы управления аккаунтом, а для администраторов или модераторов доступны ссылки на административные функции. Также код подключает необходимые стили и скрипты с помощью вызовов методов `AppAsset::register()` и `$this->head()`. В целом, данный код представляет общий шаблон для внешнего вида и навигации веб-приложения, который будет использоваться для всех страниц в приложении.

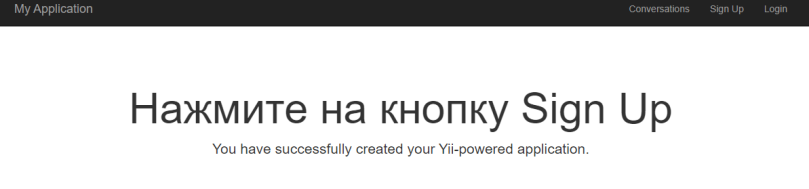


Рис.14. сама главная страница.

Дальше у меня сделана views/site где у меня прописан интерфейс регистрации и логина.

<?= $form->field($model, 'username')->textInput(['autofocus' => true]) ?>  
  
<?= $form->field($model, 'password')->passwordInput() ?>  
  
<?= $form->field($model, 'rememberMe')->checkbox([  
 'template' => "<div class=\"col-lg-offset-1 col-lg-3\">{input} {label}</div>\n<div class=\"col-lg-8\">{error}</div>",  
]) ?>

Рис.14. Код регистрации интерфейс

Вот так выглядит их интерфейс:

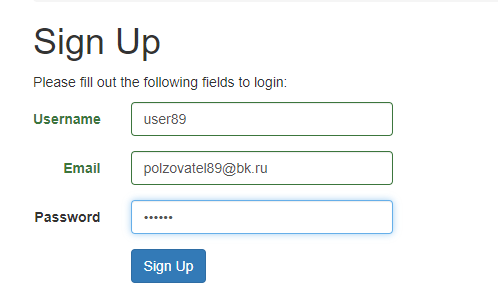


Рис.16 Интерфейс регистрации

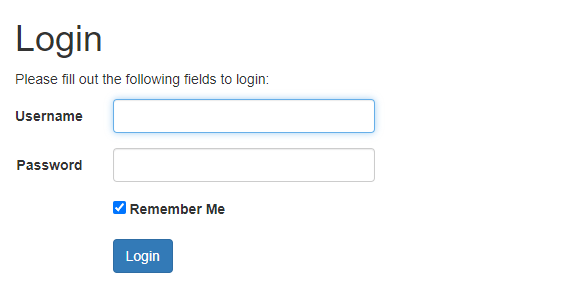


Рис.17 Интерфейс логина

Так выглядит код профиля пользователя.

<h1><?= $model->getName() ?></h1>  
  
 <?php if (!($model->user->isLogged())): ?>  
 <div class="btn-group">  
 <?php if ($user->hasFollower($model->user\_id)): ?>  
 <?= Html::*a*('Unfollow', ['user/unfollow', 'id' => $model->user\_id], ['class' => 'btn btn-danger']) ?>  
 <?php else: ?>  
 <?= Html::*a*('Follow', ['user/follow', 'id' => $model->user\_id], ['class' => 'btn btn-success']) ?>  
 <?php endif; ?>  
  
 <?= Html::*a*('Send message', ['conversation/get-conversation', 'id' => $model->user\_id], ['class' => 'btn btn-info']) ?>  
 </div>  
 <?php else: ?>  
 <?= Html::*a*('Settings', ['user/update', 'id' => $model->user\_id], ['class' => 'btn btn-danger']) ?>  
 <?php endif; ?>  
</div>

Рис.18 Код интерфейса профиля.

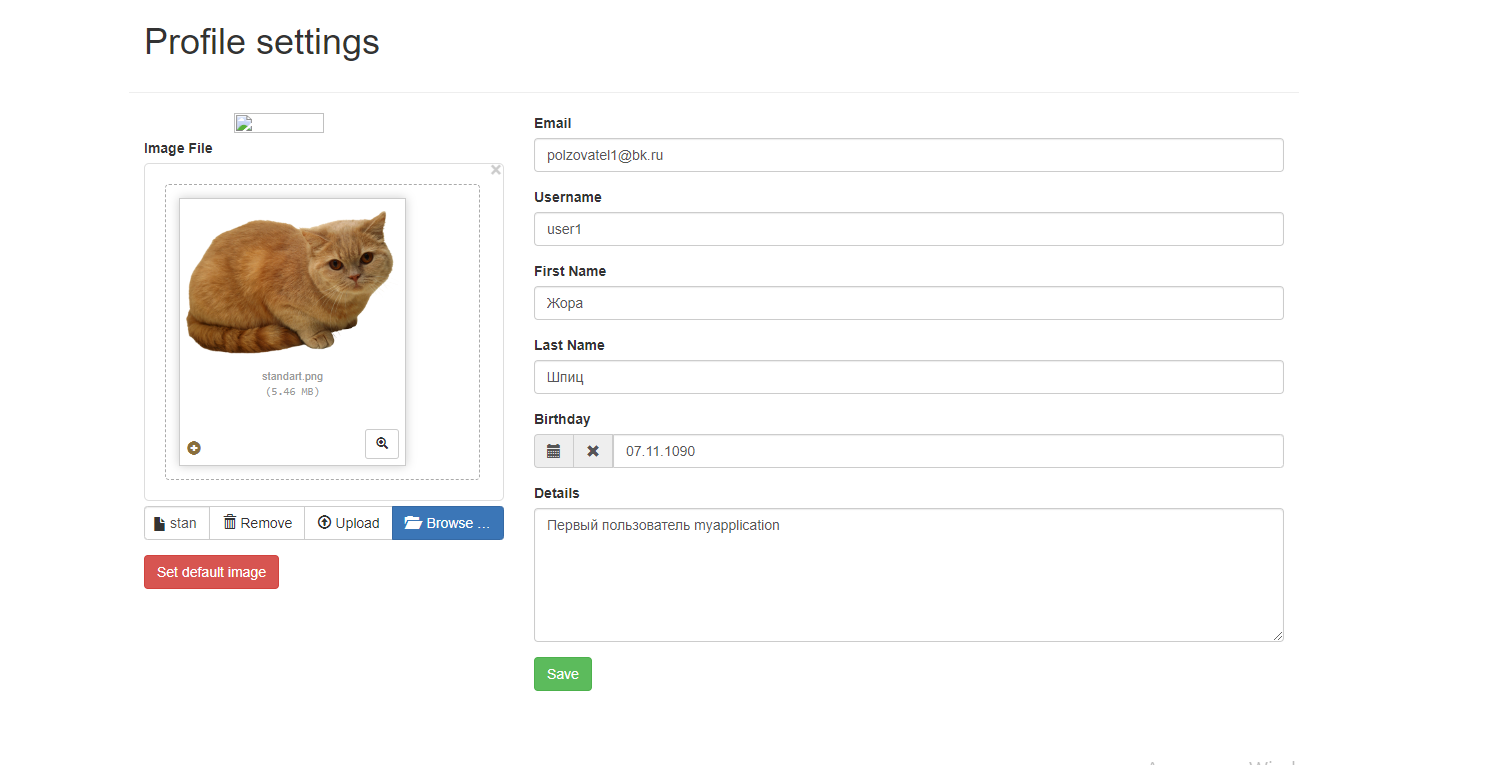


Рис.19 Интерфейс профиля

После создания интерфейса профиля я решил создать интерфейс чата, где пользователь может не только переписываться лично с другим пользователем, но создавать групповые чаты.

<div class="message pull-left well well-sm col-md-8 ">  
  
 <p class="message-author">  
 <?= Html::*a*(Html::*encode*($model->user->username), ['user/view', 'id' => $model->user->id]) ?>  
 </p>  
 <?= nl2br(Html::*encode*($model->content)) ?>  
  
 <p class="create-at text-muted pull-right"><?= Yii::*$app*->formatter->asDatetime($model->created\_at, 'short') ?></p>  
</div>  
<div class="clearfix"></div>

Рис.20 Код интерфейса чата

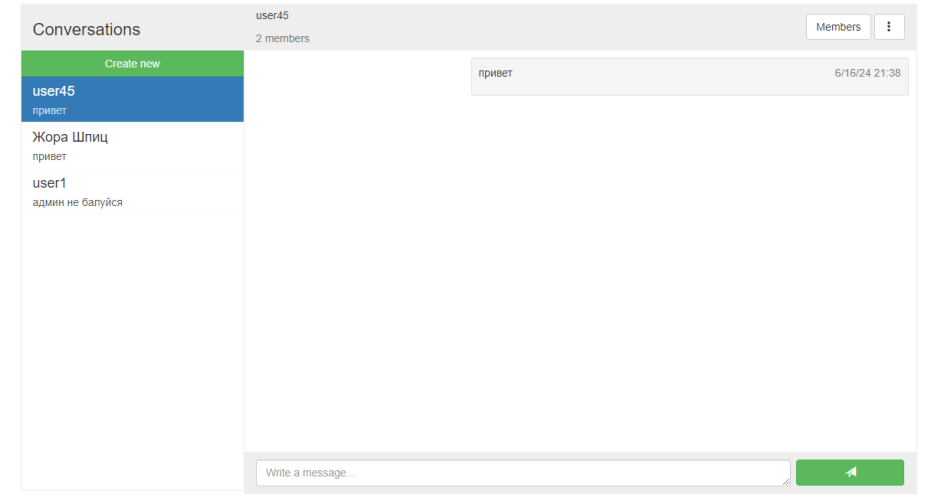


Рис. 21 Интерфейс чата.

**Заключение**

В ходе выполнения курсового проекта была разработана социальная сеть с минимальным набором функций необходимых для ее функционирования. Основной целью данного проекта было создание интерфейса для пользователя, позволяя ему добавлять в друзей и других пользователей этой социальной сети, редактирования данных и писания в чат.

Были решены задачи:

1. Выбрал стиль для своего курсового проекта, который будет приятен глазу.
2. Изучение Yii. При помощи замечательного фреймворка как Yiiframework я создал полностью функциональную социальную сеть. Подробно изучая миграции я смог связать их между собой и также связать вместе с контроллерами и моделями, которые в свою очередь выполняли определенные правила и функции.

В результате выполненного курсового проекта была достигнута основная задача- разработка социальной сети с необходимым функционалом. Все поставленные задачи были выполнены. Хотелось бы добавить, что данная курсовая работа пошла мне исключительно на пользу. Она позволила мне проявить себя и мою фантазию. Полученные результаты демонстрируют успешное выполнение проекта и подтверждают возможность дальнейшего и многократного расширения в будущем.

**Источники**

Миграции / [Электронный ресурс] // yiiframework : [сайт]. — URL: https://www.yiiframework.com/doc/guide/2.0/ru (дата обращения: 17.06.2024).

Работа с формами / [Электронный ресурс] // yiiframework : [сайт]. — URL: https://www.yiiframework.com/doc/guide/2.0/ru (дата обращения: 17.06.2024).

Модели / [Электронный ресурс] // yiiframework : [сайт]. — URL: https://www.yiiframework.com/doc/guide/2.0/ru (дата обращения: 17.06.2024).

Контроллеры / [Электронный ресурс] // yiiframework : [сайт]. — URL: https://www.yiiframework.com/doc/guide/2.0/ru (дата обращения: 17.06.2024).

O Yii / [Электронный ресурс] // yiiframework : [сайт]. — URL: https://www.yiiframework.com/doc/guide/2.0/ru (дата обращения: 17.06.2024).

Работа с паролями / [Электронный ресурс] // yiiframework : [сайт]. — URL: https://www.yiiframework.com/doc/guide/2.0/ru (дата обращения: 17.06.2024).

Фильтры / [Электронный ресурс] // yiiframework : [сайт]. — URL: https://www.yiiframework.com/doc/guide/2.0/ru (дата обращения: 17.06.2024).

Виджеты / [Электронный ресурс] // yiiframework : [сайт]. — URL: https://www.yiiframework.com/doc/guide/2.0/ru (дата обращения: 17.06.2024).

/ [Электронный ресурс] // stackoverflow : [сайт]. — URL: https://stackoverflow.com/ (дата обращения: 17.06.2024).

iPynch Simple-PHP-Social-Network-Website / iPynch [Электронный ресурс] // GitHub : [сайт]. — URL: https://github.com/iPynch/Simple-PHP-Social-Network-Website (дата обращения: 17.06.2024

Weismann yii2-social-network / Weismann [Электронный ресурс] // GitHub : [сайт]. — URL: https://github.com/Weismann9/yii2-social-network (дата обращения: 17.06.2024).

**Приложения**



Рис.1.установка Git

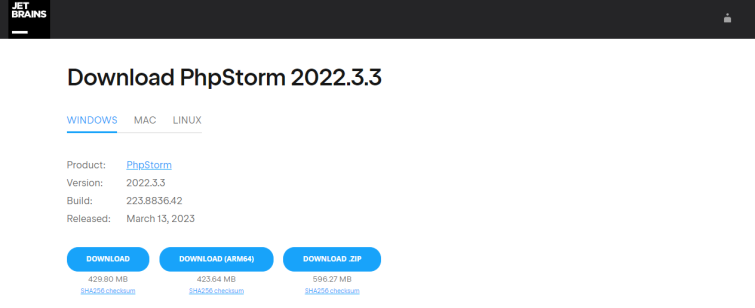


Рис.2.установка PhpStorm

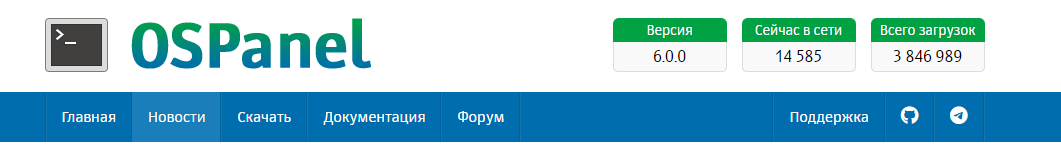


Рис.4. установка OSPanel

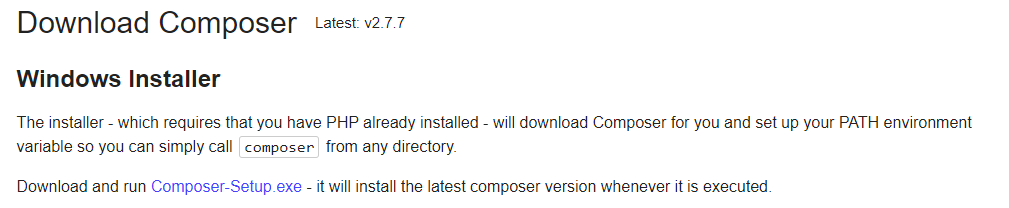


Рис.5.установка Composer

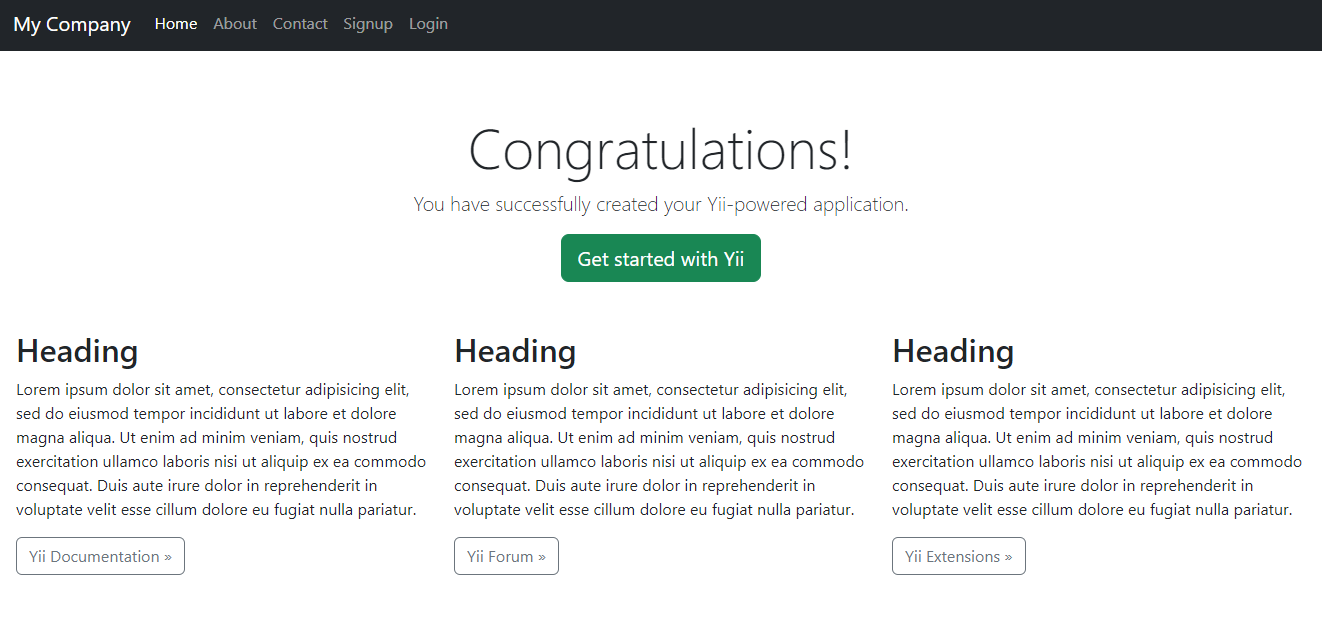


Рис.6 установка Yii

public function getGallery()  
{  
 return $this->hasOne(Gallery::*className*(), ['id' => 'gallery\_id']);  
}  
  
*/\*\*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/*public function getUser()  
{  
 return $this->hasOne(User::*className*(), ['id' => 'user\_id']);  
}  
  
public function getName()  
{  
 return isset($this->first\_name) ? $this->getFullName() : $this->getUsername();  
}  
  
*/\*\*  
 \** ***@return*** *string  
 \*/*public function getFullName()  
{  
 return $this->first\_name . ' ' . $this->last\_name;  
}  
  
*/\*\*  
 \** ***@return*** *string  
 \*/*public function getUsername()  
{  
 return $this->user->username;  
}  
  
public function beforeSave($insert)  
{  
 if (parent::*beforeSave*($insert)) {  
 if ($imageFile = UploadedFile::*getInstance*($this, 'imageFile')) {  
 $this->avatar = '/uploads/' . $imageFile->baseName . '.' . $imageFile->extension;  
 $imageFile->saveAs(substr($this->avatar, 1));  
 }  
 return true;  
 }  
 return false;  
}  
  
public function setDefaultImage()  
{  
 $this->avatar = '/gallery/standart.png';  
}

рис.7. небольшие правила работы модели

public function safeUp()  
{  
 $this->createTable('friendship', [  
 'id' => $this->primaryKey(11)->unsigned(),  
 'first\_user\_id' => $this->integer(11)->unsigned(),  
 'second\_user\_id' => $this->integer(11)->unsigned(),  
 ]);  
  
 $this->createIndex(  
 'idx-friendship-users\_id',  
 'friendship',  
 [  
 'first\_user\_id',  
 'second\_user\_id'  
 ],  
 true  
 );  
  
 // add foreign key for table `user`  
 $this->addForeignKey(  
 'fk-friendship-first\_user\_id',  
 'friendship',  
 'first\_user\_id',  
 'user',  
 'id',  
 'CASCADE',  
 'CASCADE'  
 );  
  
 // add foreign key for table `user`  
 $this->addForeignKey(  
 'fk-friendship-second\_user\_id',  
 'friendship',  
 'second\_user\_id',  
 'user',  
 'id',  
 'CASCADE',  
 'CASCADE'  
 );  
}

Рис.8 показ миграции добавления в друзья

public function rules()  
{  
 return [  
 [['first\_user\_id', 'second\_user\_id'], 'integer'],  
 [['first\_user\_id', 'second\_user\_id'], 'unique', 'targetAttribute' => ['first\_user\_id', 'second\_user\_id']],  
 [['first\_user\_id'], 'exist', 'skipOnError' => true, 'targetClass' => User::*className*(), 'targetAttribute' => ['first\_user\_id' => 'id']],  
 [['second\_user\_id'], 'exist', 'skipOnError' => true, 'targetClass' => User::*className*(), 'targetAttribute' => ['second\_user\_id' => 'id']],  
 ];  
}  
  
*/\*\*  
 \* {****@inheritdoc****}  
 \*/*public function attributeLabels()  
{  
 return [  
 'id' => 'ID',  
 'first\_user\_id' => 'First User ID',  
 'second\_user\_id' => 'Second User ID',  
 ];  
}

Рис. 9. показ модели добавления друзья

public function safeUp()  
{  
 $this->createTable('conversation', [  
 'id' => $this->primaryKey(11)->unsigned(),  
 'title' => $this->string(64),  
 ]);  
  
 $this->createTable('user\_has\_conversation', [  
 'id' => $this->primaryKey(11)->unsigned(),  
 'user\_id' => $this->integer(11)->unsigned(),  
 'conversation\_id' => $this->integer(11)->unsigned()  
 ]);

Рис.10 Демонстрация миграции

public function behaviors()  
{  
 return [  
 'access' => [  
 'class' => AccessControl::*className*(),  
 'only' => ['logout', 'login', 'signup'],  
 'rules' => [  
 [  
 'actions' => ['logout'],  
 'allow' => true,  
 'roles' => ['@'],  
 ],  
 [  
 'allow' => true,  
 'actions' => ['login', 'signup'],  
 'roles' => ['?'],  
 ],  
 ],  
 ],  
 'verbs' => [  
 'class' => VerbFilter::*className*(),  
 'actions' => [  
 'logout' => ['post'],  
 ],  
 ],  
 ];  
}public function actions()  
{  
 return [  
 'error' => [  
 'class' => 'yii\web\ErrorAction',  
 ],  
 'captcha' => [  
 'class' => 'yii\captcha\CaptchaAction',  
 'fixedVerifyCode' => *YII\_ENV\_TEST* ? 'testme' : null,  
 ],  
 ];  
}  
  
*string  
 \*/*public function actionIndex()  
{  
 return $this->render('index');  
}  
  
public function actionLogin()  
{  
 if (!Yii::*$app*->user->isGuest) {  
 return $this->goHome();  
 }  
  
 $model = new LoginForm();  
 if ($model->load(Yii::*$app*->request->post()) && $model->login()) {  
 return $this->goBack();  
 }  
  
 $model->password = '';  
 return $this->render('login', [  
 'model' => $model,  
 ]);  
}

Рис.11 Демонстрация контроллера чата.

<?php  
  
namespace app\models;  
  
use Yii;  
use yii\behaviors\TimestampBehavior;  
use yii\db\ActiveRecord;  
  
*/\*\*  
 \* This is the model class for table "conversation\_message".  
 \*  
 \** ***@property*** *string $id  
 \** ***@property*** *string $conversation\_id  
 \** ***@property*** *string $user\_id  
 \** ***@property*** *string $content  
 \** ***@property*** *string $created\_at  
 \*  
 \** ***@property*** *Conversation $conversation  
 \** ***@property*** *User $user  
 \*/*class ConversationMessage extends ActiveRecord  
{  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public static function tableName()  
 {  
 return 'conversation\_message';  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function rules()  
 {  
 return [  
 [['conversation\_id', 'user\_id'], 'integer'],  
 [['content'], 'string'],  
 [['created\_at'], 'string', 'max' => 32],  
 [['conversation\_id'], 'exist', 'skipOnError' => true, 'targetClass' => Conversation::*className*(), 'targetAttribute' => ['conversation\_id' => 'id']],  
 [['user\_id'], 'exist', 'skipOnError' => true, 'targetClass' => User::*className*(), 'targetAttribute' => ['user\_id' => 'id']],  
 ];  
 }  
  
 public function behaviors()  
 {  
 return [  
 'timestamp' => [  
 'class' => TimestampBehavior::*className*(),  
 'attributes' => [  
 ActiveRecord::*EVENT\_BEFORE\_INSERT* => ['created\_at'],  
 ],  
 'value' => date('Y-m-d H:i:s'),  
 ],  
 ];  
 }  
  
 */\*\*  
 \* {****@inheritdoc****}  
 \*/* public function attributeLabels()  
 {  
 return [  
 'id' => 'ID',  
 'conversation\_id' => 'Conversation ID',  
 'user\_id' => 'User ID',  
 'content' => 'Content',  
 'created\_at' => 'Created At',  
 ];  
 }  
  
 */\*\*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* public function getConversation()  
 {  
 return $this->hasOne(Conversation::*className*(), ['id' => 'conversation\_id']);  
 }  
  
 public function getAuthor()  
 {  
 return $this->user->getUsername();  
 }  
  
 */\*\*  
 \** ***@param*** *User $user  
 \** ***@return*** *bool  
 \*/* public function isOwner($user)  
 {  
 if ($user === $this->getUser()) {  
 return true;  
 }  
 if ($user === null) {  
 return false;  
 }  
 if ($this->getUser() === null) {  
 return false;  
 }  
 return $user->getId() === $this->user->getId();  
 }  
  
 */\*\*  
 \** ***@return*** *\yii\db\ActiveQuery  
 \*/* public function getUser()  
 {  
 return $this->hasOne(User::*className*(), ['id' => 'user\_id']);  
 }  
}

Рис.12 Демонстрация модели чата

<?php $this->beginPage() ?>  
<!DOCTYPE html>  
<html lang="<?= Yii::*$app*->language ?>">  
<head>  
 <meta charset="<?= Yii::*$app*->charset ?>">  
 <meta http-equiv="X-UA-Compatible" content="IE=edge">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <?= Html::*csrfMetaTags*() ?>  
 <title><?= Html::*encode*($this->title) ?></title>  
 <?php $this->head() ?>  
</head>  
<body>  
<?php $this->beginBody() ?>  
  
<div class="wrap">

Рис.13 Показ главной страницы.

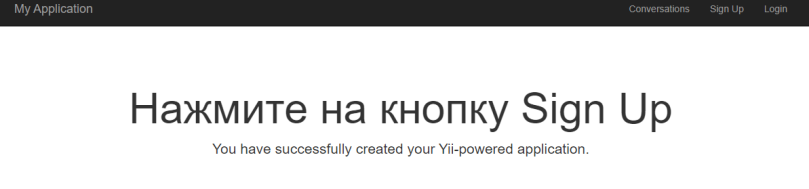


Рис.14. сама главная страница.

<?= $form->field($model, 'username')->textInput(['autofocus' => true]) ?>  
  
<?= $form->field($model, 'password')->passwordInput() ?>  
  
<?= $form->field($model, 'rememberMe')->checkbox([  
 'template' => "<div class=\"col-lg-offset-1 col-lg-3\">{input} {label}</div>\n<div class=\"col-lg-8\">{error}</div>",  
]) ?>

Рис.15. Код регистрации интерфейс

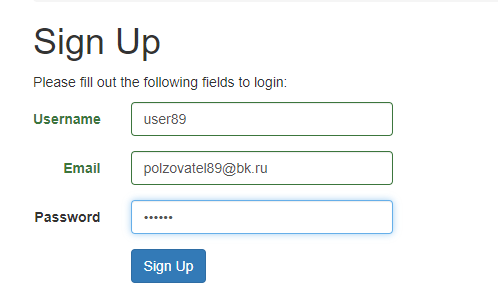


Рис.16 Интерфейс регистрации

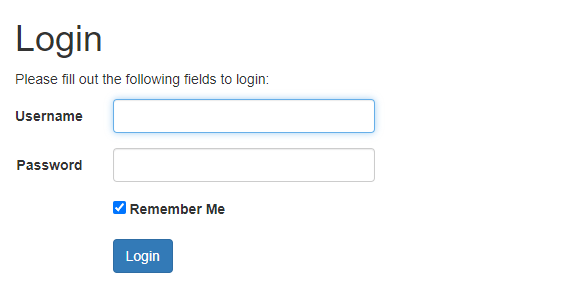


Рис.17 Интерфейс логина

<h1><?= $model->getName() ?></h1>  
  
 <?php if (!($model->user->isLogged())): ?>  
 <div class="btn-group">  
 <?php if ($user->hasFollower($model->user\_id)): ?>  
 <?= Html::*a*('Unfollow', ['user/unfollow', 'id' => $model->user\_id], ['class' => 'btn btn-danger']) ?>  
 <?php else: ?>  
 <?= Html::*a*('Follow', ['user/follow', 'id' => $model->user\_id], ['class' => 'btn btn-success']) ?>  
 <?php endif; ?>  
  
 <?= Html::*a*('Send message', ['conversation/get-conversation', 'id' => $model->user\_id], ['class' => 'btn btn-info']) ?>  
 </div>  
 <?php else: ?>  
 <?= Html::*a*('Settings', ['user/update', 'id' => $model->user\_id], ['class' => 'btn btn-danger']) ?>  
 <?php endif; ?>  
</div>

Рис.18 Код интерфейса профиля.

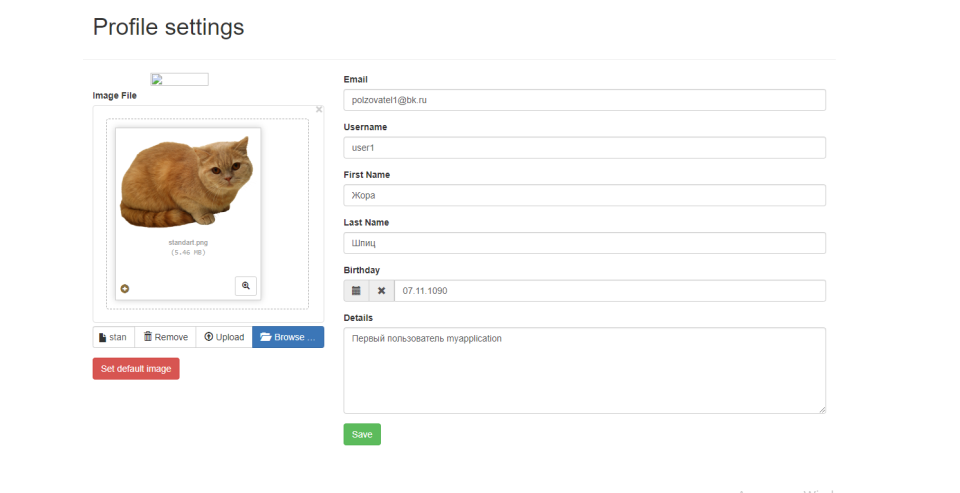


Рис.19 Интерфейс профиля

<div class="message pull-left well well-sm col-md-8 ">  
  
 <p class="message-author">  
 <?= Html::*a*(Html::*encode*($model->user->username), ['user/view', 'id' => $model->user->id]) ?>  
 </p>  
 <?= nl2br(Html::*encode*($model->content)) ?>  
  
 <p class="create-at text-muted pull-right"><?= Yii::*$app*->formatter->asDatetime($model->created\_at, 'short') ?></p>  
</div>  
<div class="clearfix"></div>

Рис.20 Код интерфейса чата

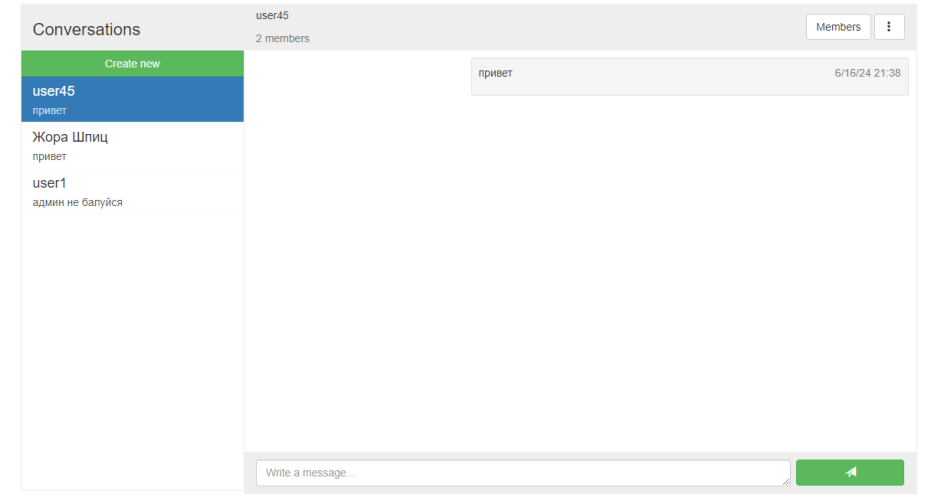


Рис. 21 Интерфейс чата.