**Сервис регистрации пользователей и администраторов**

**Описание проекта**

Простой веб-сервис для управления пользователями и администраторами с минималистичным дизайном и анимированными ошибками валидации.

**Используемые технологии**

Язык программирования: Go (Golang)

Веб-фреймворк: gorilla/mux (маршрутизация)

Управление сессиями: gorilla/sessions

Безопасность: bcrypt (хэширование паролей)

База данных: MySQL (через драйвер github.com/go-sql-driver/mysql)

Верстка: HTML5 + CSS3 с встроенными стилями

Анимации: CSS transitions и keyframe animations

**Основные функции**

**Для пользователей:**

* Регистрация с валидацией (логин ≥5 символов, пароль ≥8 символов)
* Вход в систему с проверкой учетных данных
* Личный кабинет с отображением имени пользователя
* Выход из аккаунта

**Для администраторов:**

* Автоматическое создание первого админа при первом запуске (логин: admin, пароль: admin123)
* Панель администратора с возможностью создания новых админов
* Разграничение прав доступа по ролям (user/admin)

**Безопасность:**

* Хэширование паролей с помощью bcrypt
* Уникальность логинов
* Защита маршрутов через middleware проверки сессий
* Валидация входных данных

**Дизайн и UX:**

* Минималистичный интерфейс с градиентными фонами
* Анимированные сообщения об ошибках (slide-in и shake эффекты)
* Адаптивная верстка
* Визуальная обратная связь при успешных действиях

**Структура проекта**

* Единый файл main.go с полной логикой приложения
* Папка templates/ с HTML-шаблонами (4 страницы)
* Автоматическая инициализация базы данных и первого администратора

**Запуск**

* Создать базу данных user\_admin\_service в XAMPP MySQL
* Установить зависимости через go mod
* Запустить сервер командой go run main.go
* Открыть в браузере: http://localhost:8080

**Листинг программы**

**main.go**

package **main**

import (

    "crypto/rand"

    "database/sql"

    "encoding/hex"

    "fmt"

    "html/template"

    "log"

    "net/http"

    "regexp"

    "github.com/gorilla/mux"

    "github.com/gorilla/sessions"

    "golang.org/x/crypto/bcrypt"

    \_ "github.com/go-sql-driver/mysql"

)

type **User** struct {

    ID       int

    Username string

    Role     string

}

var (

    db       \***sql**.**DB**

    store    \***sessions**.**CookieStore**

    templates map[string]\***template**.**Template**

)

*//компилирует HTML шаблоны*

func **initTemplates**() {

    templates = **make**(map[string]\***template**.**Template**)

    templates["register"] = template.**Must**(template.**ParseFiles**("templates/register.html"))

    templates["login"] = template.**Must**(template.**ParseFiles**("templates/login.html"))

    templates["user"] = template.**Must**(template.**ParseFiles**("templates/user.html"))

    templates["admin"] = template.**Must**(template.**ParseFiles**("templates/admin.html"))

}

*//подключается к MySQL базе данных*

func **connectDB**() \***sql**.**DB** {

*// По умолчанию: пользователь 'root', без пароля, база 'user\_admin\_service'*

    dsn := "root:@tcp(127.0.0.1:3306)/user\_admin\_service?charset=utf8mb4&parseTime=True&loc=Local"

    db, err := sql.**Open**("mysql", dsn)

    if err != nil {

        log.**Fatal**("Ошибка подключения к базе данных:", err)

    }

    if err := db.**Ping**(); err != nil {

        log.**Fatal**("Ошибка проверки подключения к базе данных:", err)

    }

    log.**Println**("Успешное подключение к базе данных")

    return db

}

*//создает первого администратора, если он не существует*

func **createFirstAdmin**() {

    var count int

    err := db.**QueryRow**("SELECT COUNT(\*) FROM users WHERE role = 'admin'").**Scan**(&count)

    if err != nil {

        log.**Fatal**("Ошибка проверки существования админа:", err)

    }

    if count == 0 {

*// Хэш пароль*

        passwordHash, err := bcrypt.**GenerateFromPassword**([]byte("admin123"), bcrypt.DefaultCost)

        if err != nil {

            log.**Fatal**("Ошибка хэширования пароля админа:", err)

        }

        \_, err = db.**Exec**("INSERT INTO users (username, password\_hash, role) VALUES (?, ?, ?)",

            "admin", string(passwordHash), "admin")

        if err != nil {

            log.**Fatal**("Ошибка создания первого админа:", err)

        }

        log.**Println**("Создан первый администратор: логин='admin', пароль='admin123'")

    }

}

*//валидность логина*

func **validateUsername**(username string) bool {

    if **len**(username) < 5 {

        return false

    }

    matched, \_ := regexp.**MatchString**("^[a-zA-Z0-9\_-]+$", username)

    return matched

}

*//валидность пароля*

func **validatePassword**(password string) bool {

    return **len**(password) >= 8

}

*//получает пользователя по логину*

func **getUserByUsername**(username string) (\***User**, error) {

    user := &**User**{}

    err := db.**QueryRow**("SELECT id, username, role FROM users WHERE username = ?", username).**Scan**(

        &user.ID, &user.Username, &user.Role)

    if err != nil {

        if err == sql.ErrNoRows {

            return nil, nil

        }

        return nil, err

    }

    return user, nil

}

*//создает нового пользователя*

func **createUser**(username, password, role string) error {

*//существует ли уже такой логин*

    existingUser, err := **getUserByUsername**(username)

    if err != nil {

        return err

    }

    if existingUser != nil {

        return fmt.**Errorf**("пользователь с таким логином уже существует")

    }

*//Хэш пароль*

    passwordHash, err := bcrypt.**GenerateFromPassword**([]byte(password), bcrypt.DefaultCost)

    if err != nil {

        return err

    }

    \_, err = db.**Exec**("INSERT INTO users (username, password\_hash, role) VALUES (?, ?, ?)",

        username, string(passwordHash), role)

    return err

}

*//проверяет учетные данные пользователя*

func **authenticate**(username, password string) (\***User**, error) {

    var passwordHash string

    var user **User**

    err := db.**QueryRow**("SELECT id, username, role, password\_hash FROM users WHERE username = ?", username).**Scan**(

        &user.ID, &user.Username, &user.Role, &passwordHash)

    if err != nil {

        if err == sql.ErrNoRows {

            return nil, fmt.**Errorf**("неверный логин или пароль")

        }

        return nil, err

    }

*// Проверяем пароль*

    err = bcrypt.**CompareHashAndPassword**([]byte(passwordHash), []byte(password))

    if err != nil {

        return nil, fmt.**Errorf**("неверный логин или пароль")

    }

    return &user, nil

}

*//случайный ключ для сессии*

func **generateSessionKey**() string {

    bytes := **make**([]byte, 32)

    rand.**Read**(bytes)

    return hex.**EncodeToString**(bytes)

}

*//получает пользователя из сессии*

func **getUserFromSession**(r \***http**.**Request**) (\***User**, error) {

    session, \_ := store.**Get**(r, "user-session")

    userID, ok := session.Values["user\_id"].(int)

    if !ok {

        return nil, fmt.**Errorf**("пользователь не авторизован")

    }

    var user **User**

    err := db.**QueryRow**("SELECT id, username, role FROM users WHERE id = ?", userID).**Scan**(

        &user.ID, &user.Username, &user.Role)

    if err != nil {

        return nil, err

    }

    return &user, nil

}

*// requireAuth middleware для проверки аутентификации*

func **requireAuth**(next **http**.**HandlerFunc**) **http**.**HandlerFunc** {

    return func(w **http**.**ResponseWriter**, r \***http**.**Request**) {

        session, \_ := store.**Get**(r, "user-session")

        userID, ok := session.Values["user\_id"].(int)

        if !ok {

            http.**Redirect**(w, r, "/login", http.StatusSeeOther)

            return

        }

*// Проверяем, что пользователь существует в базе*

        var count int

        err := db.**QueryRow**("SELECT COUNT(\*) FROM users WHERE id = ?", userID).**Scan**(&count)

        if err != nil || count == 0 {

            http.**Redirect**(w, r, "/login", http.StatusSeeOther)

            return

        }

**next**(w, r)

    }

}

*// requireAdmin middleware для проверки прав администратора*

func **requireAdmin**(next **http**.**HandlerFunc**) **http**.**HandlerFunc** {

    return func(w **http**.**ResponseWriter**, r \***http**.**Request**) {

        session, \_ := store.**Get**(r, "user-session")

        userID, ok := session.Values["user\_id"].(int)

        if !ok {

            http.**Redirect**(w, r, "/login", http.StatusSeeOther)

            return

        }

*// Проверяем роль пользователя*

        var role string

        err := db.**QueryRow**("SELECT role FROM users WHERE id = ?", userID).**Scan**(&role)

        if err != nil || role != "admin" {

            http.**Redirect**(w, r, "/user", http.StatusSeeOther)

            return

        }

**next**(w, r)

    }

}

*//обработка страниц*

func **registerHandler**(w **http**.**ResponseWriter**, r \***http**.**Request**) {

    if r.Method == "POST" {

        username := r.**FormValue**("username")

        password := r.**FormValue**("password")

        confirmPassword := r.**FormValue**("confirm\_password")

*// Валидация*

        errors := []string{}

        if !**validateUsername**(username) {

            errors = **append**(errors, "Логин должен содержать не менее 5 символов и состоять только из английских букв, цифр, \_ и -")

        }

        if !**validatePassword**(password) {

            errors = **append**(errors, "Пароль должен содержать не менее 8 символов")

        }

        if password != confirmPassword {

            errors = **append**(errors, "Пароли не совпадают")

        }

        if **len**(errors) > 0 {

            data := map[string]interface{}{

                "Username": username,

                "Errors":   errors,

            }

            templates["register"].**Execute**(w, data)

            return

        }

*// Создаем юзера*

        err := **createUser**(username, password, "user")

        if err != nil {

            errors = **append**(errors, err.**Error**())

            data := map[string]interface{}{

                "Username": username,

                "Errors":   errors,

            }

            templates["register"].**Execute**(w, data)

            return

        }

*//перенаправляем на страницу входа*

        http.**Redirect**(w, r, "/login?registered=true", http.StatusSeeOther)

        return

    }

*// GET запрос - показываем форму регистрации*

    templates["register"].**Execute**(w, nil)

}

func **loginHandler**(w **http**.**ResponseWriter**, r \***http**.**Request**) {

    if r.Method == "POST" {

        username := r.**FormValue**("username")

        password := r.**FormValue**("password")

        user, err := **authenticate**(username, password)

        if err != nil {

            data := map[string]interface{}{

                "Username": username,

                "Errors":   []string{err.**Error**()},

            }

            templates["login"].**Execute**(w, data)

            return

        }

*// Создаем сессию*

        session, \_ := store.**Get**(r, "user-session")

        session.Values["user\_id"] = user.ID

        session.**Save**(r, w)

*// Перенаправляем в зависимости от роли*

        if user.Role == "admin" {

            http.**Redirect**(w, r, "/admin", http.StatusSeeOther)

        } else {

            http.**Redirect**(w, r, "/user", http.StatusSeeOther)

        }

        return

    }

*// GET запрос - показываем форму входа*

    registered := r.URL.**Query**().**Get**("registered")

    data := map[string]interface{}{

        "Registered": registered == "true",

    }

    templates["login"].**Execute**(w, data)

}

func **userHandler**(w **http**.**ResponseWriter**, r \***http**.**Request**) {

    user, err := **getUserFromSession**(r)

    if err != nil {

        http.**Redirect**(w, r, "/login", http.StatusSeeOther)

        return

    }

    data := map[string]interface{}{

        "User": user,

    }

    templates["user"].**Execute**(w, data)

}

func **adminHandler**(w **http**.**ResponseWriter**, r \***http**.**Request**) {

    user, err := **getUserFromSession**(r)

    if err != nil || user.Role != "admin" {

        http.**Redirect**(w, r, "/login", http.StatusSeeOther)

        return

    }

    if r.Method == "POST" {

        username := r.**FormValue**("username")

        password := r.**FormValue**("password")

        confirmPassword := r.**FormValue**("confirm\_password")

        errors := []string{}

        if !**validateUsername**(username) {

            errors = **append**(errors, "Логин должен содержать не менее 5 символов и состоять только из английских букв, цифр, \_ и -")

        }

        if !**validatePassword**(password) {

            errors = **append**(errors, "Пароль должен содержать не менее 8 символов")

        }

        if password != confirmPassword {

            errors = **append**(errors, "Пароли не совпадают")

        }

        if **len**(errors) > 0 {

            data := map[string]interface{}{

                "User":   user,

                "Errors": errors,

            }

            templates["admin"].**Execute**(w, data)

            return

        }

*// Создаем администратора*

        err := **createUser**(username, password, "admin")

        if err != nil {

            errors = **append**(errors, err.**Error**())

            data := map[string]interface{}{

                "User":   user,

                "Errors": errors,

            }

            templates["admin"].**Execute**(w, data)

            return

        }

*// Успешное создание*

        data := map[string]interface{}{

            "User":        user,

            "Success":     "Администратор успешно создан!",

        }

        templates["admin"].**Execute**(w, data)

        return

    }

*// GET запрос - показываем панель администратора*

    templates["admin"].**Execute**(w, map[string]interface{}{"User": user})

}

func **logoutHandler**(w **http**.**ResponseWriter**, r \***http**.**Request**) {

    session, \_ := store.**Get**(r, "user-session")

    session.Options.MaxAge = -1

    session.**Save**(r, w)

    http.**Redirect**(w, r, "/login", http.StatusSeeOther)

}

func **main**() {

*// Подключаемся к базе данных*

    db = **connectDB**()

    defer db.**Close**()

*// Создаем первого администратора*

**createFirstAdmin**()

*// Инициализируем сессии*

    store = sessions.**NewCookieStore**([]byte(**generateSessionKey**()))

*// Компилируем шаблоны*

**initTemplates**()

*// Создаем маршрутизатор*

    r := mux.**NewRouter**()

*// Публичные маршруты*

    r.**HandleFunc**("/register", registerHandler).**Methods**("GET", "POST")

    r.**HandleFunc**("/login", loginHandler).**Methods**("GET", "POST")

    r.**HandleFunc**("/logout", logoutHandler).**Methods**("GET")

*// Защищенные маршруты*

    r.**HandleFunc**("/user", **requireAuth**(userHandler)).**Methods**("GET")

    r.**HandleFunc**("/admin", **requireAdmin**(**requireAuth**(adminHandler))).**Methods**("GET", "POST")

    log.**Println**("Сервер запущен на http://localhost:8080")

    log.**Fatal**(http.**ListenAndServe**(":8080", r))

}

user.html

<!DOCTYPE *html*>

<html *lang*="ru">

<head>

    <meta *charset*="UTF-8">

    <meta *name*="viewport" *content*="width=device-width, initial-scale=1.0">

    <title>Личный кабинет</title>

    <style>

        \* {

            margin: 0;

            padding: 0;

            box-sizing: border-box;

            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

        }

        body {

            background: **linear-gradient**(135deg, #f093fb 0%, #f5576c 100%);

            min-height: 100vh;

            display: flex;

            justify-content: center;

            align-items: center;

            padding: 20px;

        }

        .container {

            background: white;

            padding: 40px;

            border-radius: 15px;

            box-shadow: 0 15px 35px **rgba**(0, 0, 0, 0.1);

            width: 100%;

            max-width: 450px;

            text-align: center;

        }

        h1 {

            margin-bottom: 30px;

            color: #333;

            font-size: 28px;

            font-weight: 600;

        }

        .user-info {

            background: #f8f9fa;

            padding: 20px;

            border-radius: 10px;

            margin-bottom: 30px;

        }

        .user-info h2 {

            color: #666;

            margin-bottom: 10px;

            font-size: 20px;

        }

        .user-info p {

            color: #444;

            font-size: 18px;

            font-weight: 500;

        }

        .logout-btn {

            padding: 12px 30px;

            background: **linear-gradient**(135deg, #f093fb 0%, #f5576c 100%);

            color: white;

            border: none;

            border-radius: 8px;

            font-size: 16px;

            font-weight: 600;

            cursor: pointer;

            transition: transform 0.2s ease;

            text-decoration: none;

            display: inline-block;

        }

        .logout-btn:hover {

            transform: **translateY**(-2px);

        }

    </style>

</head>

<body>

    <div *class*="container">

        <h1>Личный кабинет</h1>

        <div *class*="user-info">

            <h2>Добро пожаловать,</h2>

            <p>{{.User.Username}}</p>

        </div>

        <a *href*="/logout" *class*="logout-btn">Выйти из аккаунта</a>

    </div>

</body>

</html>

register.html

<!DOCTYPE *html*>

<html *lang*="ru">

<head>

    <meta *charset*="UTF-8">

    <meta *name*="viewport" *content*="width=device-width, initial-scale=1.0">

    <title>Регистрация</title>

    <style>

        \* {

            margin: 0;

            padding: 0;

            box-sizing: border-box;

            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

        }

        body {

            background: **linear-gradient**(135deg, #667eea 0%, #764ba2 100%);

            min-height: 100vh;

            display: flex;

            justify-content: center;

            align-items: center;

            padding: 20px;

        }

        .container {

            background: white;

            padding: 40px;

            border-radius: 15px;

            box-shadow: 0 15px 35px **rgba**(0, 0, 0, 0.1);

            width: 100%;

            max-width: 450px;

        }

        h1 {

            text-align: center;

            margin-bottom: 30px;

            color: #333;

            font-size: 28px;

            font-weight: 600;

        }

        .form-group {

            margin-bottom: 20px;

        }

        label {

            display: block;

            margin-bottom: 8px;

            color: #555;

            font-weight: 500;

        }

        input[type="text"],

        input[type="password"] {

            width: 100%;

            padding: 12px 15px;

            border: 2px solid #e1e1e1;

            border-radius: 8px;

            font-size: 16px;

            transition: border-color 0.3s ease;

        }

        input[type="text"]:focus,

        input[type="password"]:focus {

            outline: none;

            border-color: #667eea;

        }

        .btn {

            width: 100%;

            padding: 12px;

            background: **linear-gradient**(135deg, #667eea 0%, #764ba2 100%);

            color: white;

            border: none;

            border-radius: 8px;

            font-size: 16px;

            font-weight: 600;

            cursor: pointer;

            transition: transform 0.2s ease;

        }

        .btn:hover {

            transform: **translateY**(-2px);

        }

        .error-messages {

            background: #fee;

            border: 1px solid #fcc;

            border-radius: 8px;

            padding: 15px;

            margin-bottom: 20px;

            animation: slideIn 0.3s ease forwards;

        }

        .error-item {

            color: #d32f2f;

            margin-bottom: 5px;

            animation: shake 0.5s ease;

        }

        .error-item:last-child {

            margin-bottom: 0;

        }

        .login-link {

            text-align: center;

            margin-top: 20px;

            color: #667eea;

            text-decoration: none;

            font-weight: 500;

            transition: color 0.2s ease;

        }

        .login-link:hover {

            color: #764ba2;

        }

        @keyframes slideIn {

            from {

                opacity: 0;

                transform: **translateY**(-10px);

            }

            to {

                opacity: 1;

                transform: **translateY**(0);

            }

        }

        @keyframes shake {

            0%, 100% { transform: **translateX**(0); }

            25% { transform: **translateX**(-3px); }

            75% { transform: **translateX**(3px); }

        }

    </style>

</head>

<body>

    <div *class*="container">

        <h1>Регистрация</h1>

        {{if .Errors}}

        <div *class*="error-messages">

            {{range .Errors}}

            <div *class*="error-item">{{.}}</div>

            {{end}}

        </div>

        {{end}}

        <form *method*="POST">

            <div *class*="form-group">

                <label *for*="username">Логин</label>

                <input *type*="text" *id*="username" *name*="username" *value*="{{.Username}}" *required*>

            </div>

            <div *class*="form-group">

                <label *for*="password">Пароль</label>

                <input *type*="password" *id*="password" *name*="password" *required*>

            </div>

            <div *class*="form-group">

                <label *for*="confirm\_password">Подтвердите пароль</label>

                <input *type*="password" *id*="confirm\_password" *name*="confirm\_password" *required*>

            </div>

            <button *type*="submit" *class*="btn">Зарегистрироваться</button>

        </form>

        <a *href*="/login" *class*="login-link">Уже есть аккаунт? Войти</a>

    </div>

</body>

</html>

login.html

<!DOCTYPE *html*>

<html *lang*="ru">

<head>

    <meta *charset*="UTF-8">

    <meta *name*="viewport" *content*="width=device-width, initial-scale=1.0">

    <title>Вход</title>

    <style>

        \* {

            margin: 0;

            padding: 0;

            box-sizing: border-box;

            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

        }

        body {

            background: **linear-gradient**(135deg, #4facfe 0%, #00f2fe 100%);

            min-height: 100vh;

            display: flex;

            justify-content: center;

            align-items: center;

            padding: 20px;

        }

        .container {

            background: white;

            padding: 40px;

            border-radius: 15px;

            box-shadow: 0 15px 35px **rgba**(0, 0, 0, 0.1);

            width: 100%;

            max-width: 450px;

        }

        h1 {

            text-align: center;

            margin-bottom: 30px;

            color: #333;

            font-size: 28px;

            font-weight: 600;

        }

        .success-message {

            background: #e8f5e8;

            border: 1px solid #c8e6c9;

            border-radius: 8px;

            padding: 15px;

            margin-bottom: 20px;

            color: #2e7d32;

            animation: slideIn 0.3s ease forwards;

        }

        .form-group {

            margin-bottom: 20px;

        }

        label {

            display: block;

            margin-bottom: 8px;

            color: #555;

            font-weight: 500;

        }

        input[type="text"],

        input[type="password"] {

            width: 100%;

            padding: 12px 15px;

            border: 2px solid #e1e1e1;

            border-radius: 8px;

            font-size: 16px;

            transition: border-color 0.3s ease;

        }

        input[type="text"]:focus,

        input[type="password"]:focus {

            outline: none;

            border-color: #4facfe;

        }

        .btn {

            width: 100%;

            padding: 12px;

            background: **linear-gradient**(135deg, #4facfe 0%, #00f2fe 100%);

            color: white;

            border: none;

            border-radius: 8px;

            font-size: 16px;

            font-weight: 600;

            cursor: pointer;

            transition: transform 0.2s ease;

        }

        .btn:hover {

            transform: **translateY**(-2px);

        }

        .error-messages {

            background: #fee;

            border: 1px solid #fcc;

            border-radius: 8px;

            padding: 15px;

            margin-bottom: 20px;

            animation: slideIn 0.3s ease forwards;

        }

        .error-item {

            color: #d32f2f;

            margin-bottom: 5px;

            animation: shake 0.5s ease;

        }

        .register-link {

            text-align: center;

            margin-top: 20px;

            color: #4facfe;

            text-decoration: none;

            font-weight: 500;

            transition: color 0.2s ease;

        }

        .register-link:hover {

            color: #00f2fe;

        }

        @keyframes slideIn {

            from {

                opacity: 0;

                transform: **translateY**(-10px);

            }

            to {

                opacity: 1;

                transform: **translateY**(0);

            }

        }

        @keyframes shake {

            0%, 100% { transform: **translateX**(0); }

            25% { transform: **translateX**(-3px); }

            75% { transform: **translateX**(3px); }

        }

    </style>

</head>

<body>

    <div *class*="container">

        <h1>Вход в систему</h1>

        {{if $.Registered}}

        <div *class*="success-message">

            Регистрация прошла успешно! Теперь вы можете войти в систему.

        </div>

        {{end}}

        {{if .Errors}}

        <div *class*="error-messages">

            {{range .Errors}}

            <div *class*="error-item">{{.}}</div>

            {{end}}

        </div>

        {{end}}

        <form *method*="POST">

            <div *class*="form-group">

                <label *for*="username">Логин</label>

                <input *type*="text" *id*="username" *name*="username" *value*="{{.Username}}" *required*>

            </div>

            <div *class*="form-group">

                <label *for*="password">Пароль</label>

                <input *type*="password" *id*="password" *name*="password" *required*>

            </div>

            <button *type*="submit" *class*="btn">Войти</button>

        </form>

        <a *href*="/register" *class*="register-link">Нет аккаунта? Зарегистрироваться</a>

    </div>

</body>

</html>

admin.html

<!DOCTYPE *html*>

<html *lang*="ru">

<head>

    <meta *charset*="UTF-8">

    <meta *name*="viewport" *content*="width=device-width, initial-scale=1.0">

    <title>Панель администратора</title>

    <style>

        \* {

            margin: 0;

            padding: 0;

            box-sizing: border-box;

            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

        }

        body {

            background: **linear-gradient**(135deg, #43e97b 0%, #38f9d7 100%);

            min-height: 100vh;

            display: flex;

            justify-content: center;

            align-items: center;

            padding: 20px;

        }

        .container {

            background: white;

            padding: 40px;

            border-radius: 15px;

            box-shadow: 0 15px 35px **rgba**(0, 0, 0, 0.1);

            width: 100%;

            max-width: 500px;

        }

        h1 {

            text-align: center;

            margin-bottom: 30px;

            color: #333;

            font-size: 28px;

            font-weight: 600;

        }

        .admin-info {

            background: #e8f5f5;

            padding: 15px;

            border-radius: 8px;

            margin-bottom: 30px;

            text-align: center;

        }

        .admin-info h2 {

            color: #2e7d7d;

            margin-bottom: 5px;

            font-size: 18px;

        }

        .admin-info p {

            color: #1b5e5e;

            font-weight: 500;

        }

        .success-message {

            background: #e8f5e8;

            border: 1px solid #c8e6c9;

            border-radius: 8px;

            padding: 15px;

            margin-bottom: 20px;

            color: #2e7d32;

            animation: slideIn 0.3s ease forwards;

        }

        .form-group {

            margin-bottom: 20px;

        }

        label {

            display: block;

            margin-bottom: 8px;

            color: #555;

            font-weight: 500;

        }

        input[type="text"],

        input[type="password"] {

            width: 100%;

            padding: 12px 15px;

            border: 2px solid #e1e1e1;

            border-radius: 8px;

            font-size: 16px;

            transition: border-color 0.3s ease;

        }

        input[type="text"]:focus,

        input[type="password"]:focus {

            outline: none;

            border-color: #43e97b;

        }

        .btn {

            width: 100%;

            padding: 12px;

            background: **linear-gradient**(135deg, #43e97b 0%, #38f9d7 100%);

            color: white;

            border: none;

            border-radius: 8px;

            font-size: 16px;

            font-weight: 600;

            cursor: pointer;

            transition: transform 0.2s ease;

        }

        .btn:hover {

            transform: **translateY**(-2px);

        }

        .error-messages {

            background: #fee;

            border: 1px solid #fcc;

            border-radius: 8px;

            padding: 15px;

            margin-bottom: 20px;

            animation: slideIn 0.3s ease forwards;

        }

        .error-item {

            color: #d32f2f;

            margin-bottom: 5px;

            animation: shake 0.5s ease;

        }

        .logout-link {

            text-align: center;

            margin-top: 20px;

            color: #43e97b;

            text-decoration: none;

            font-weight: 500;

            transition: color 0.2s ease;

            display: block;

        }

        .logout-link:hover {

            color: #38f9d7;

        }

        @keyframes slideIn {

            from {

                opacity: 0;

                transform: **translateY**(-10px);

            }

            to {

                opacity: 1;

                transform: **translateY**(0);

            }

        }

        @keyframes shake {

            0%, 100% { transform: **translateX**(0); }

            25% { transform: **translateX**(-3px); }

            75% { transform: **translateX**(3px); }

        }

    </style>

</head>

<body>

    <div *class*="container">

        <h1>Панель администратора</h1>

        <div *class*="admin-info">

            <h2>Вы вошли как администратор</h2>

            <p>{{.User.Username}}</p>

        </div>

        {{if .Success}}

        <div *class*="success-message">

            {{.Success}}

        </div>

        {{end}}

        {{if .Errors}}

        <div *class*="error-messages">

            {{range .Errors}}

            <div *class*="error-item">{{.}}</div>

            {{end}}

        </div>

        {{end}}

        <h3>Создать нового администратора</h3>

        <form *method*="POST">

            <div *class*="form-group">

                <label *for*="username">Логин</label>

                <input *type*="text" *id*="username" *name*="username" *required*>

            </div>

            <div *class*="form-group">

                <label *for*="password">Пароль</label>

                <input *type*="password" *id*="password" *name*="password" *required*>

            </div>

            <div *class*="form-group">

                <label *for*="confirm\_password">Подтвердите пароль</label>

                <input *type*="password" *id*="confirm\_password" *name*="confirm\_password" *required*>

            </div>

            <button *type*="submit" *class*="btn">Создать администратора</button>

        </form>

        <a *href*="/logout" *class*="logout-link">Выйти из аккаунта</a>

    </div>

</body>

</html>