Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Северо-Осетинский государственный университет

имени Коста Левановича Хетагурова»

**КУРСОВАЯ РАБОТА**

«Система голосования для IT-куба»

Выполнил:

студент 2 курса направления

«Прикладная математика и информатика»

Гуссаов Георгий Асламбекович

Научный руководитель:

старший преподаватель кафедры

прикладной математики и информатики

Макаренко М.Д.

Владикавказ 2024

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ 3

ГЛАВА 1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 4

* 1. Функционал 4
  2. Этапы реализации 4

ГЛАВА 2. ПРАКТИЧЕСКАЯ ЧАСТЬ 5

2.1 Инструкция пользователя 5

2.2 Инструкция системного администратора 10

2.3 Инструкция программиста 10

ЗАКЛЮЧЕНИЕ 16

СПИСОК ЛИТЕРАТУРЫ 17

**ВВЕДЕНИЕ**

Программа Яндекс лицея – это фундаментальная двухлетняя программа по Python с плавным погружением: от изучения основ и синтаксиса до разработки веб-приложений. Раз в год в IT-кубе, который реализует программу Яндекс лицея, проводится Python-парад. Этот парад проводится для:

1. для привлечения новых учеников с помощью демонстрации приложений, разработанных учениками в процессе обучения
2. для получения обратная связи между разработчиками (учениками) и пользователями: другими учениками, преподавателями, родителями
3. создание здоровой конкурентной среды, для мотивации разработчиков (учеников)

Чтобы решить все эти задачи при проведении Python-парада используется технология голосования. Из всех проектов каждый участник парада должен выбрать три лучших в трех номинациях: самый сложный, самый красивый, самый прикольный. Ранее использовалась система голосования Google Form, однако она обладает целым рядом отрицательных свойств:

* нельзя проголосовать только за три проекта
* сложно добавить описание проектов, за которые голосуешь, чтобы не промахнуться
* система никак не защищена от накрутки голосов

Таким образом, возникла необходимость разработки системы голосования, не имеющей всех вышеперечисленных недостатков.

**Постановка задачи**

Разработать систему голосования для Python- парада, проводимом в IT-кубе, которая:

* будет веб-приложением
* позволяет добавлять несколько парадов
* позволяет однократно голосовать пользователю с телефона
* голосовать только в разработанном центром ДО формате
* хранить данные голосования
* формировать результаты голосования в виде диаграммы
* защита от накручивания голосов пользователями

**1. Техническое задание**

1.1. Функционал голосующего.

1. Форма для регистрации пользователя (сохранение в куки его данных).

2. Форма для голосования.

3. Просмотр результатов голосования, после их публикации.

4. Просмотр таймера до закрытия голосования.

1.2. Функционал разработчика.

1. Форма для регистрации проекта (доступна по ссылке).

2. Форма для редактирования проекта.

1.3. Функционал админа.

1. Форма для авторизации админа(сохранение в куки его данных).

2. Страница со списком всех событий.

3. Форма для добавления события.

4. Форма для регистрации проекта.

5. Функционал для каждого события:

* Просмотр страницы с голосами.
* Просмотр страницы с результатами.
* Просмотр страницы со списком проектов.
* Редактирование и удаления проекта.

6. Админ может голосовать неограниченное кол-во раз.

**2. Этапы реализации**

1. Версия 0.1

* Выводить форму для регистрации пользователя.
* После прохождения регистрации сама форма голосования.

1. Версия 0.2

* Выводить форму для авторизации админа.
* После авторизации админа показывать страницу с датами событий.
* Админ может добавить новую дату.
* Админ может добавлять и удалять проекты.

1. Версия 0.3

* Реализация функционала для каждой даты (для админа).
* Реализовать сохранение в куки пользователя.
* Реализовать сохранение в куки админа.

**Глава 1. Практическая часть**

При разработке приложения были приняты следующие решения:

1. Приложение каждым пользователем используется только один раз, поэтому нет необходимости вводить регистрацию пользователя. С другой стороны, организаторам парадов, необходима статистика по посетителям. Поэтому пользователю достаточно ввести свою фамилию и имя, а также класс, поскольку парады проводятся школьниками для школьников. Но посетителями бывают родители и учителя, поэтому необходимо добавить соответствующий выбор
2. Итоги парада проводятся по трем номинациям: самый сложный, самый красивый, самый прикольный проект. Каждый пользователь в каждой номинации выбрать только один проект. Один проект может быть только в одной номинации.
3. Если пользователь проголосовал за проект, больше на страницу голосования он попасть не может.
4. Система голосования должна быть оптимизирована под мобильные устройства.
5. Процесс голосования происходит только в дату проведения парада и завершается по команде администратора.
6. После завершения парада результаты голосования в виде диаграммы публикуются только по команде администратора, поскольку для подведения итогов необходимо некоторое время.
7. У разработчиков проектов должен быть доступ к заполнению информации о своем проекте.
8. Администратор должен добавлять парад на определенную дату, редактировать проекты, завершать голосование и публиковать результаты голосования.

**1.1. Инструкция пользователя**

При запуске приложения В ДЕНЬ ГОЛОСОВАНИЯ пользователь видит страницу регистрации:

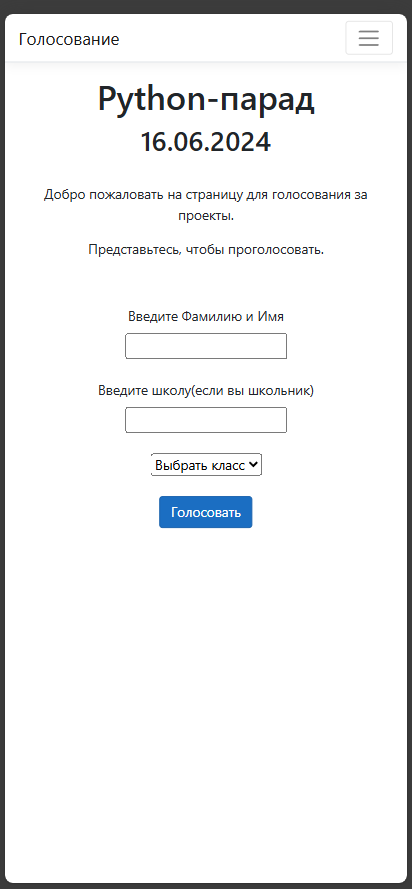


Рисунок 1.1. Страница регистрации

Если же пользователь зайдет не в день голосования, то будет выведена страница, сообщающая о том, что сегодня голосовать нельзя.

После прохождения регистрации его перекидывает на форму для голосования:

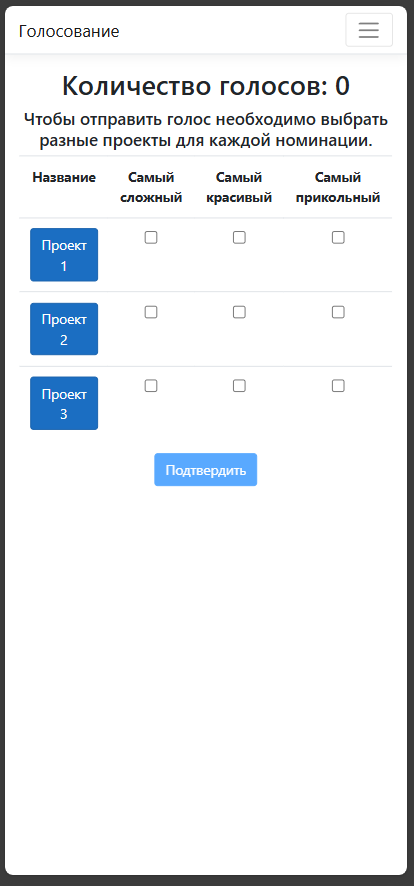


Рисунок 1.2. Форма для голосования.

После прохождения голосования пользователь переходит на страницу благодарности и в целом это весь функционал голосующего.

Рассмотрим функционал для администратора. Чтобы попасть на страницу админа надо ввести логин и пароль администратора:

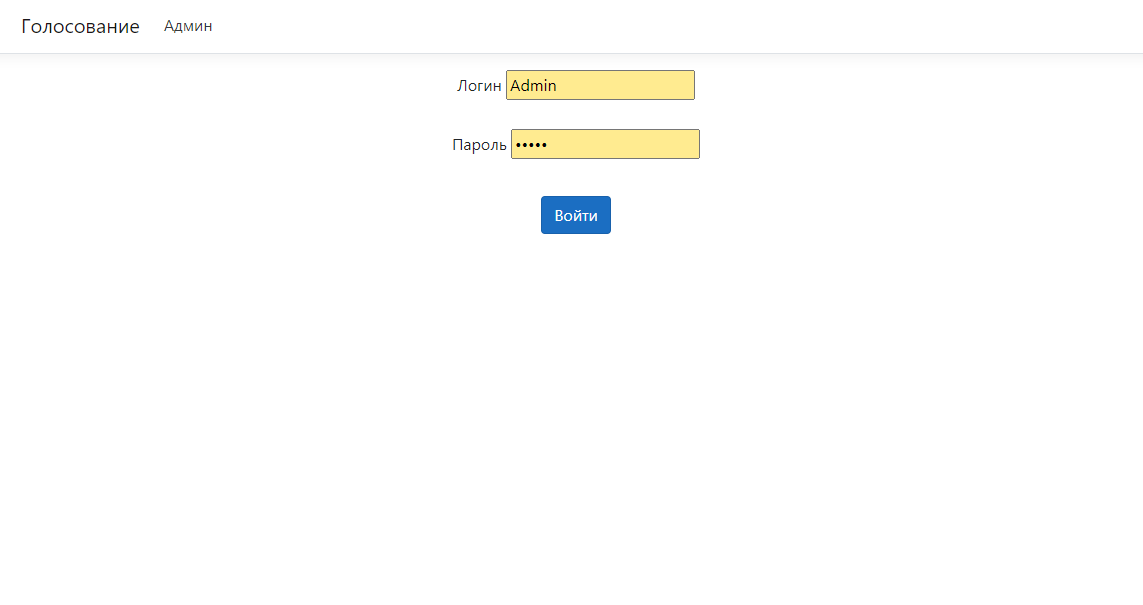


Рисунок 1.3. Форма авторизации администратора.

После авторизации администратор видит список событий:

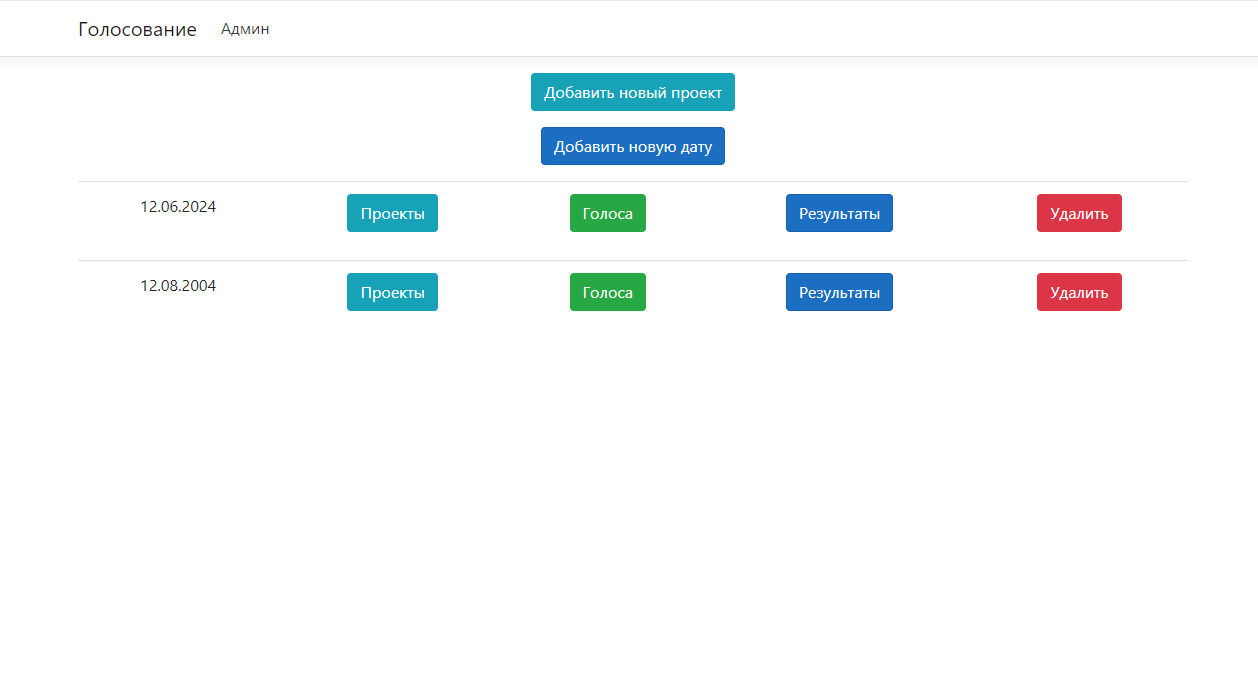


Рисунок 1.4. Главная страница администратора.

Далее в картинках покажу каждую страницу администратора:

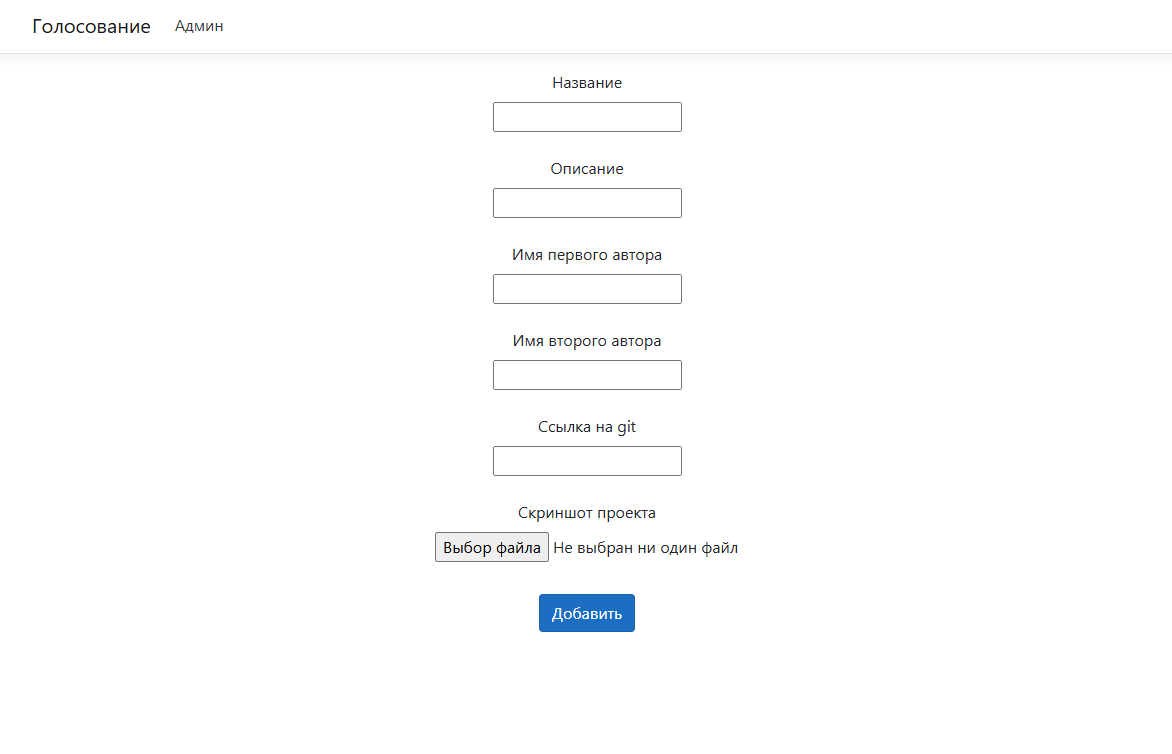


Рисунок 1.5. Страница добавления нового проекта.

Страницу добавления нового события(даты) пропустил, т.к. там буквально одно поле для заполнения даты.

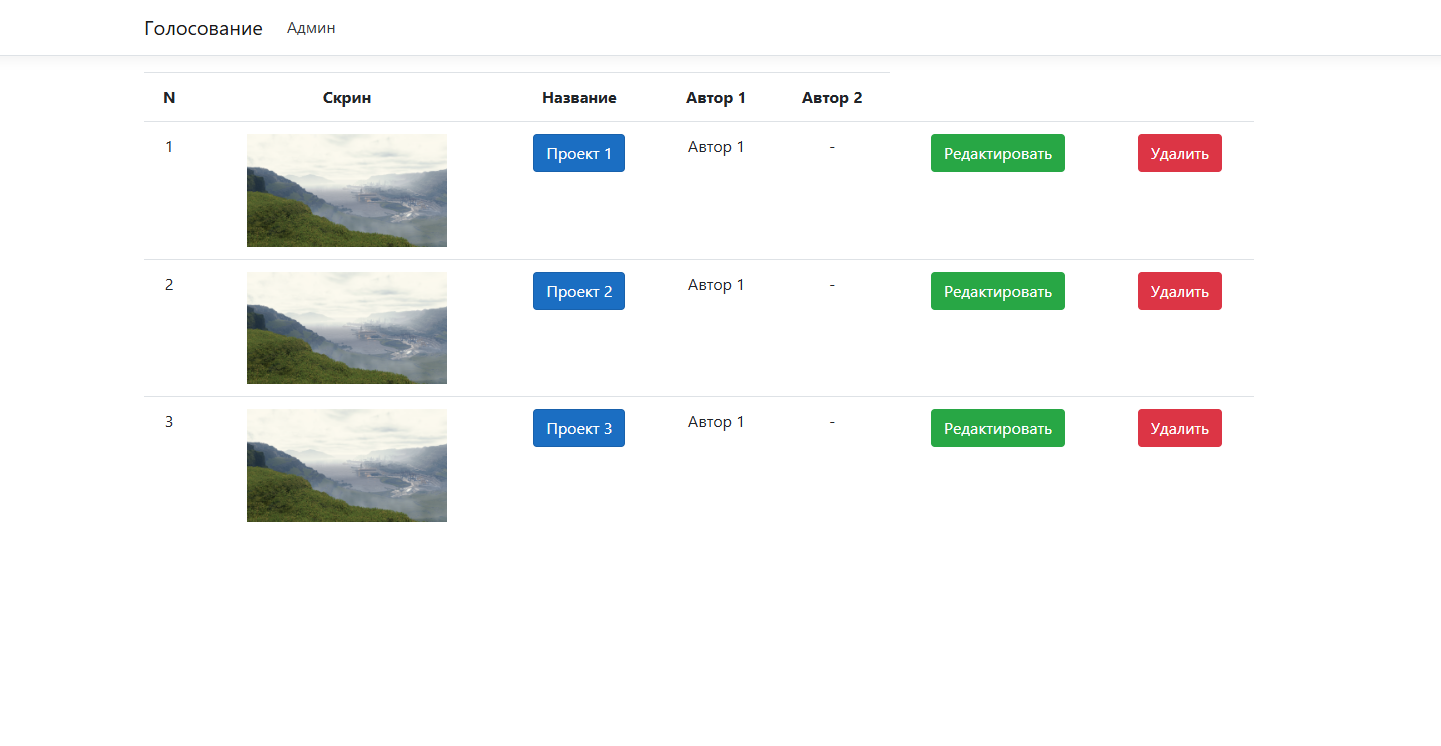


Рисунок 1.6. Страница для отображения списка проектов.

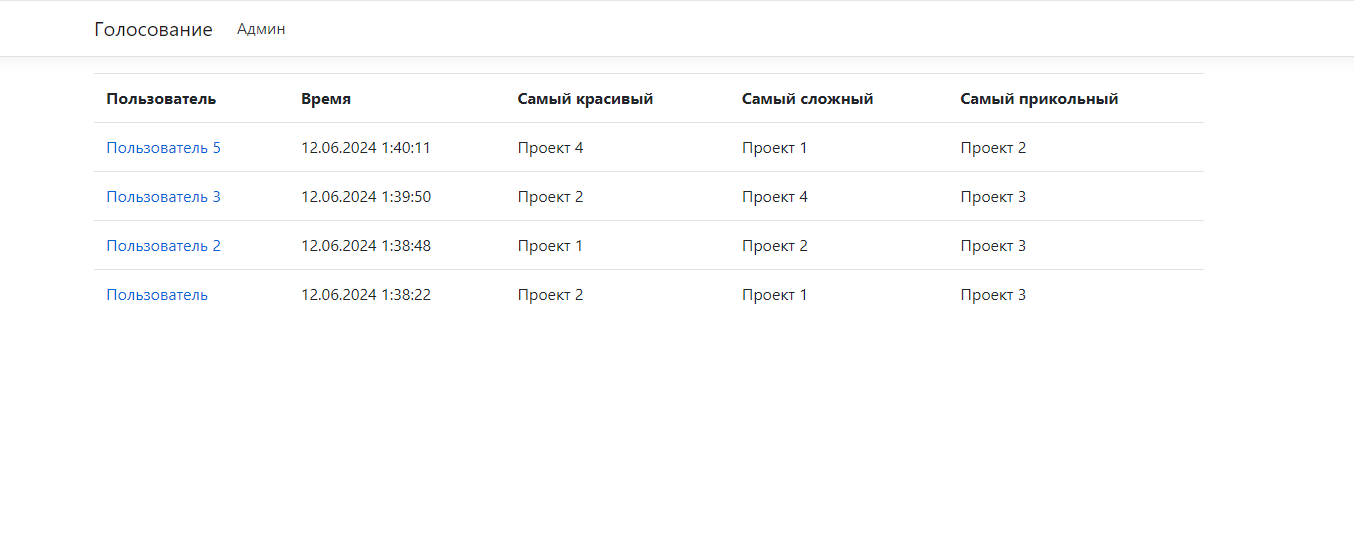


Рисунок 1.7. Страница для отображения списка голосов.

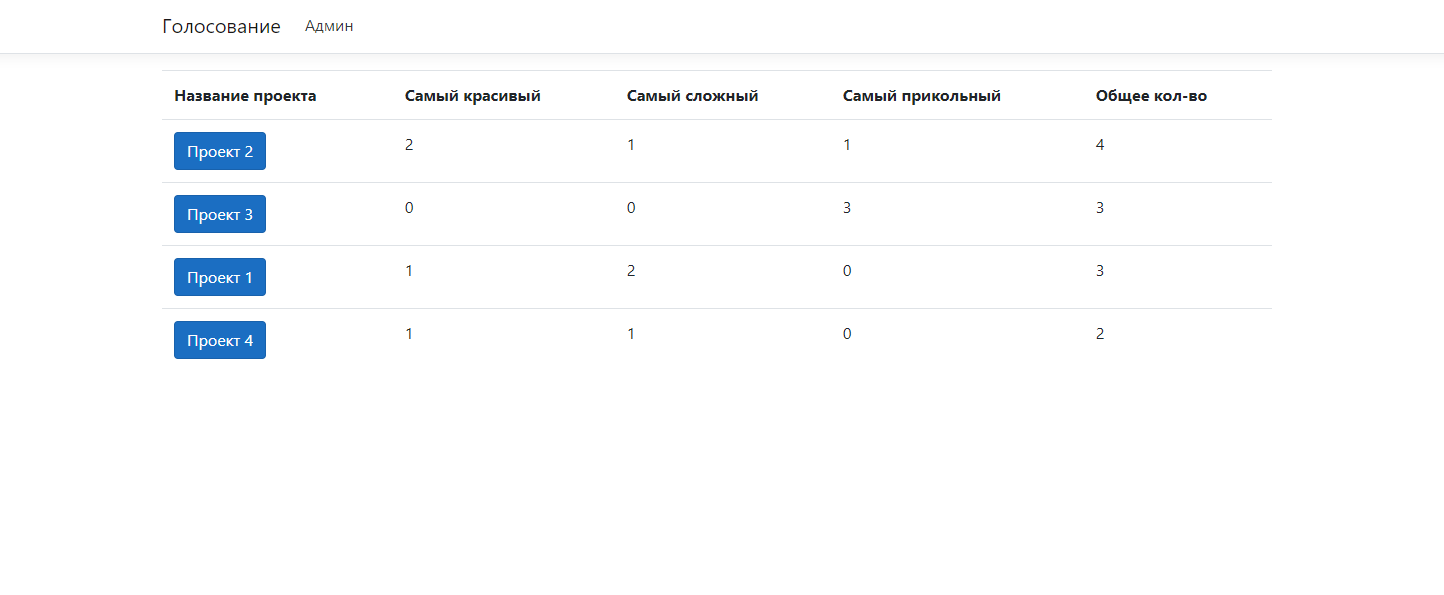


Рисунок 1.8. Страница для отображения результатов голосования.

**1.2. Инструкция системного администратора**

Требования к программным и техническим средствам для работы приложения:

* Операционная система не ниже Windows 7, Linux (с поддержкой .NET Core), macOS (с поддержкой .NET Core);
* На системе должен быть установлен .NET Core 3.1 SDK;
* Место на диске не менее 1 ГБ;
* Не менее 5,0 МБ ОЗУ;
* Любой современный браузер;

Данные для входа в аккаунт администратора:

* Логин: Admin
* Пароль Admin

**1.3. Инструкция программиста**

Для начала покажу модели для хранения информации, по которым создаются таблицы базы данных при помощи Entity Framework:

Это модель для события:

public class Event

{

public Guid Id { get; set; }

public DateTime DateTime { get; set; }

public List<Vote> Votes { get; set; }

public List<Project> Projects { get; set; }

public Event()

{

Projects = new List<Project>();

Votes = new List<Vote>();

}

}

Подробнее о ней:

Id – идентификатор события.

DateTime – дата события.

List<Vote> Votes – список голосов, сделанных в этом событии.

List<Project> Projects – список проектов, привязанных к этому событию.

Это модель для проекта:

public class Project

{

public Guid Id { get; set; }

public string Name { get; set; }

public string Description { get; set; }

public string FirsAuthor { get; set; }

public string SecondAuthor { get; set; }

public string GitLink { get; set; }

public string ImgPath { get; set; }

}

Подробнее:

Id – идентификатор проекта.

Name – название проекта.

Description – описание проекта.

FirstAuthor – имя первого автора проекта.

SecondAuthor – имя второго автора проекта.

GitLink – ссылка на репозиторий github-а, где хранится проект.

ImgPath – локальный путь к картинке, а именно скриншоту проекта.

Это модель для пользователя (голосующего):

public class User

{

public Guid Id { get; set; }

public string Name { get; set; }

public string School { get; set; }

public string Class { get; set; }

}

Подробнее:

Id – идентификатор пользователя.

Name – ФИ пользователя.

School – школа, в которой обучается (обучался) пользователь.

Class – номер класса пользователя. В случае если пользователь не школьник, то поле заполняется его ролью (родитель, преподаватель, студент).

Это модель для голоса:

public class Vote

{

public Guid Id { get; set; }

public User User { get; set; }

public DateTime Time { get; set; }

public Project MostDificult { get; set; }

public Project MostBeautiful { get; set; }

public Project Coolest { get; set; }

public Vote()

{

Time = DateTime.Now;

}

}

Подробнее:

Id – идентификатор голоса.

User – пользователь, отправивший голос.

Time – время, в которое был сделан голос.

Project MostDificult – голос за самый сложный проект.

Project MostBeautiful – голос за самый красивый проект.

Project Coolest – голос за самый прикольный проект.

Для более понятной связности таблиц в базе данных:

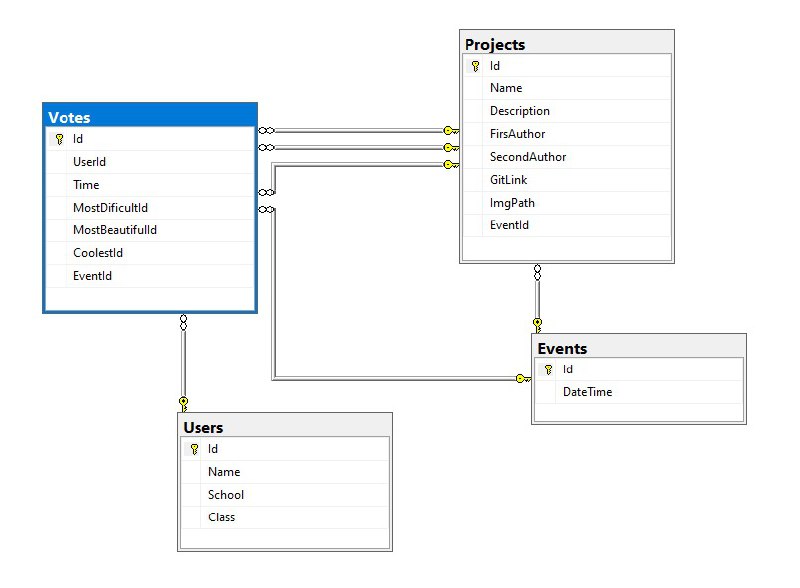


Рисунок 2.1. Диаграмма базы данных.

Далее хотелось бы рассказать про контроллеры.

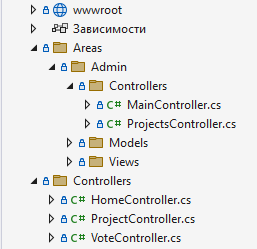


Рисунок 2.2. Контроллеры.

Для администратора я выделил отдельную область.

**MainController** – отвечает за авторизацию и отображение главной страницы администратора, а также за весь функционал на главной странице (добавление события, добавление проекта).

**ProjectsController** – отвечает за отображение списка проектов и за редактирование отдельно взятого проекта.

**HomeController** – отвечает за отображение формы регистрации пользователя.

**VoteController** – отвечает за отображение формы для голосования.

**ProjectController** –отвечает за отображение отдельно взятого проекта.

Далее покажу репозиторий событий(дат) EventsDbRepository, т.к. в базе данных все привязано к событиям и соответственно добавление новых пользователей, проектов, голосов делается через репозиторий событий.

namespace ItCubeVoteDb

{

public void AddVote(Vote vote)

{

\_dbContext.Events.OrderBy(x => x.DateTime).Include(x => x.Votes).Include(x => x.Projects).LastOrDefault().Votes.Add(vote);

\_dbContext.SaveChanges();

}

public void AddProject(Project project)

{

if (\_dbContext.Events.Count() != 0)

{

\_dbContext.Events.OrderBy(x => x.DateTime).Include(x => x.Projects).Include(x => x.Votes).LastOrDefault().Projects.Add(project);

\_dbContext.SaveChanges();

}

}

public List<Event> GetDates()

{

return \_dbContext.Events.ToList();

}

public Event GetCurrentDate()

{

if(\_dbContext.Events.Count() != 0)

{

return \_dbContext.Events.Include(x => x.Projects).Include(x => x.Votes).ThenInclude(x => x.User).OrderBy(x => x.DateTime).LastOrDefault();

}

return null;

}

public Event TryGetDateById(Guid id)

{

return \_dbContext.Events.Include(x => x.Projects).Include(x => x.Votes).Include(x => x.Votes).ThenInclude(x => x.User).FirstOrDefault(x => x.Id == id);

}

public List<Project> TryGetProjectsById(Guid id)

{

return \_dbContext.Events.Include(x => x.Projects).Include(x => x.Votes).FirstOrDefault(d => d.Id == id).Projects;

}

public void DeleteDateById(Guid id)

{

\_dbContext.Events.Remove(TryGetDateById(id));

\_dbContext.SaveChanges();

}

}

Подробнее о методах:

Add(Date date) – метод для добавления нового события.

AddVote(Vote vote) – метод для добавления голоса в список голосов события.

AddProject(Project project) – метод для добавления проекта в список проектов события.

GetDates() – метод для получения всех событий.

GetCurrentDate() – метод для получения актуального(последнего добавленного) события.

TryGetDateById(Guid id) – метод для получения события по id.

TryGetProjectsById(Guid id) – метод для получения списка проектов для определеннго события (в скобках id события).

EditProject(Guid id, Project project) – метод для изменения проекта. Id – идентификатор события, project – проект из этого события.

DeleteDateById(Guid id) – метод для удаления события по id.

**Реализация однократного голосования.**

При нажатии на начальном окне регистрации (рис. 1.1) кнопки Голосовать программа переходит на action: Login.

public IActionResult Login(UserViewModel user)

{

if (ModelState.IsValid)

{

user.Id = Guid.NewGuid();

usersDb.Add(Mapping.ToUser(user));

CookieOptions cookie = new CookieOptions();

cookie.Expires = DateTime.Now.AddYears(5);

Response.Cookies.Append("user", user.Id.ToString(), cookie);

return RedirectToAction("Index", "Vote");

}

return View("Index");

}

Если пользователь указал валидные данные, то он сохраняется сначала в таблицу пользователей в базе данных, а потом сохраняется в куки с ключом user и со значением id пользователя.

Когда же в следующий раз будет попытка подключиться к окну регистрации, то пользователя всё равно сразу будет перебрасывать на страницу голосования (либо на страницу о благодарности за участие в голосовании в случае, если он уже проголосовал). Ниже прикреплю action основной страницы и опишу его код.

public IActionResult Index()

{

if (datesDb.GetCurrentDate() != null)

{

ViewBag.CurrentDate = datesDb.GetCurrentDate().DateTime.ToShortDateString();

var Cookie = Request.Cookies["user"];

if (Cookie == null) return View();

return RedirectToAction("Index", "Vote");

}

return RedirectToAction("NoEvents");

}

В первой строке написана проверка на наличие событий, чтобы в случае их отсутствия перенаправить пользователя на соответствующую страницу. Во ViewBag я записал текущую дату, чтобы выводить её на страницу голосования.

Далее проверяется есть ли кука с названием user. Если она есть, то как я написал выше пользователя не запустит зарегистрироваться снова. Если же куки нет, то соответственно пользователь еще не регистрировался и соответственно страница для регистрации ему открывается.

**ЗАКЛЮЧЕНИЕ**

В целом можно сказать, что проект практически готов.

В программе реализовано:

1. Регистрация пользователей(голосующих) и сохранение их в куки.
2. Страница голосования.
3. Авторизация администратора и защита его страниц.
4. Добавление проектов администратором или участниками.
5. Добавление нового события администратором.
6. Возможность админа просматривать голоса и результаты голосования.

В планах на реализацию:

1. Реализовать закрытие голосования вне дня голосования.
2. Возможность админа голосовать, при чем бесконечное кол-во раз.

За время работы над этим приложением я узнал много нового и закрепил то, что уже знал. Например, научился работать с Entity Framework, ViewBag, Bootstrap. А также закрепил: JavaScript, HTML, технологию MVC.

Ссылка на проект: https://github.com/GeorgeGussaov/ItCubeVote

**СПИСОК ЛИТЕРАТУРЫ**

## 1. Andrew Lock. ASP.NET Core in Action [учебное пособие]. - Manning Publications, 2018. — 704 p. — ISBN 978-1-61729-461-3.

##### **2.** Adam Freeman. Pro ASP.NET Core 3 [учебное пособие]. - Apress, 2020. — 1088 p. — ISBN 978-1-4842-5436-9.

3. Документация по .NET Core [электронный ресурс] // URL: https://learn.microsoft.com/ru-ru/dotnet/fundamentals/

4. Полное руководство по языку программирования С# и платформе .NET. URL: <https://metanit.com/sharp/tutorial/>

5. Официальная документация по С#. URL: [Документация по C# Microsoft Docs](https://docs.microsoft.com/ru-ru/dotnet/csharp/)