**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО ЭВМ**

**ИДЗ**

**по дисциплине «Введение в нереляционные базы данных»**

**Тема: DevContest**

| Студенты гр. 2381 |  | Заметаев М. |
| --- | --- | --- |
| Студенты гр. 2381 |  | Шляхтин М. |
| Студенты гр. 2381 |  | Долотов Н. |
| Студенты гр. 2381 |  | Зазуля И. |
| Студенты гр. 2381 |  | Самулевич С. |
| Преподаватель |  | Заславский М.М. |

Санкт-Петербург

2025

**ЗАДАНИЕ**

| Студент Заметаев М.  Студент Шляхтин М.  Студент Долотов Н.  Студент Зазуля И.  Студент Самулевич С. | | |
| --- | --- | --- |
| Группа 2381 | | |
| Тема работы: разработка веб-платформа для проведения конкурсов на выполнение креативных и технических задач | | |
| Исходные данные:  Сделать фриланс–биржу, где заказчики могут организовывать конкурсы для выполнения креативных и технических задач, оценивают полученные решения и выбирают победителей, которые получают вознаграждение. Фрилансеры соревнуются между собой, стараясь предоставить наилучшее решение.  Используемая база данных – MongoDB. | | |
| Содержание пояснительной записки:  «Введение», «Сценарии использования», «Модель данных», «Разработанное приложение», «Заключение», «Приложения», «Литература» | | |
| Предполагаемый объем пояснительной записки:  Не менее 10 страниц. | | |
| Дата выдачи задания: 05.02.2025 | | |
| Дата сдачи реферата: 22.05.2025 | | |
| Дата защиты реферата: 22.05.2025   | Студенты гр. 2381 |  | Заметаев М. | | --- | --- | --- | | Студенты гр. 2381 |  | Шляхтин М. | | Студенты гр. 2381 |  | Долотов Н. | | Студенты гр. 2381 |  | Зазуля И. | | Студенты гр. 2381 |  | Самулевич С. | | Преподаватель |  | Заславский М.М. | | | |

**АННОТАЦИЯ**

В рамках работы разработана веб-платформа DevContest для проведения конкурсов на выполнение креативных и технических задач. Основная цель — упростить взаимодействие между заказчиками и фрилансерами посредством конкурсной модели. Реализация выполнена на основе клиент-серверной архитектуры с использованием React (фронтенд), Flask (бэкенд) и MongoDB (база данных). В ходе проекта были спроектированы и внедрены нереляционные структуры хранения данных, реализованы функции создания, редактирования и просмотра конкурсов, решений и отзывов. Система поддерживает регистрацию пользователей с разными ролями, импорт/экспорт данных, и развернута в контейнерах с помощью Docker.

**SUMMARY**

As part of the work, a web platform DevContest was developed to hold contests for creative and technical tasks. The main goal is to simplify the interaction between customers and freelancers through a contest model. The implementation is based on client-server architecture using React (frontend), Flask (backend) and MongoDB (database). During the project non-relational data storage structures were designed and implemented, functions of creating, editing and viewing contests, decisions and reviews were realized. The system supports user registration with different roles, data import/export, and is deployed in containers using Docker.

**СОДЕРЖАНИЕ**

| 1. | Введение | 5 |
| --- | --- | --- |
| 1.1. | Актуальность решаемой проблемы | 5 |
| 1.2. | Постановка задачи | 5 |
| 1.3. | Предлагаемое решение | 5 |
| 1.4. | Качественные требования к решению | 6 |
| 2. | Сценарии использования | 7 |
| 2.1. | Макет UI | 7 |
| 2.2. | Сценарии использования для задачи | 7 |
| 2.3. | Вывод о том, какие операции (чтение или запись) будут преобладать для вашего решения. | 10 |
| 3. | Модель данных | 11 |
| 3.1. | Нереляционная модель данных | 11 |
| 3.2.  3.3. | Реляционная модель данных  Сравнение моделей | 19  28 |
| 4. | Разработанное приложение | 29 |
| 4.1 | Краткое описание (из каких модулей / контейнеров состоит, какую архитектуру вы использовали) | 29 |
| 4.2 | Использованные технологии | 30 |
| 4.3 | Снимки экрана приложения | 32 |
|  | Заключение | 39 |
|  | Список использованных источников | 41 |
|  | Приложение А. Документация по сборке и развертыванию | 42 |
|  | Приложение Б. Инструкция для пользователя | 43 |

**1. ВВЕДЕНИЕ**

**1.1. Актуальность решаемой проблемы**

Современные компании, особенно стартапы и малый бизнес, сталкиваются с проблемой высоких затрат времени и ресурсов на поиск квалифицированных фрилансеров для выполнения креативных и технических задач, таких как разработка логотипов, веб-дизайна или прототипов приложений. Одновременно начинающие фрилансеры испытывают трудности с доступом к заказам из-за отсутствия опыта и портфолио, что замедляет их профессиональный рост. Платформы конкурсного фриланса, такие как 99designs, disignCrowd, частично решают эти проблемы, но часто имеют высокие комиссии (15–20%), сложные процессы отбора или проблемы с доступом из России, что снижает их доступность для небольших компаний и начинающих специалистов.

**1.2. Постановка задачи**

Цель работы - разработка веб-приложения DevContest — платформы для проведения конкурсов на выполнение креативных и технических задач (дизайн, разработка, иллюстрации).

Стек: React (фронтенд), Flask (бэкенд), Axios (запросы), MongoDB (данные).

Задачи: спроектировать MongoDB-схему, создать REST API, интерфейс и обеспечить CRUD-операции.

**1.3. Предлагаемое решение**

DevContest — веб-платформа с конкурсной моделью: заказчик публикует задачу с описанием, сроками и призовым фондом; фрилансеры предлагают решения; заказчик выбирает из предложенных решений победителя. В рамках проекта будет реализована платформа с использованием документно-ориентированной базы данных MongoDB. Данные хранятся в виде документов формата JSON(BSON), что обеспечивает гибкость при создании различного контента — конкурсных заданий, решений и отзывов.

**1.4. Качественные требования к решению**

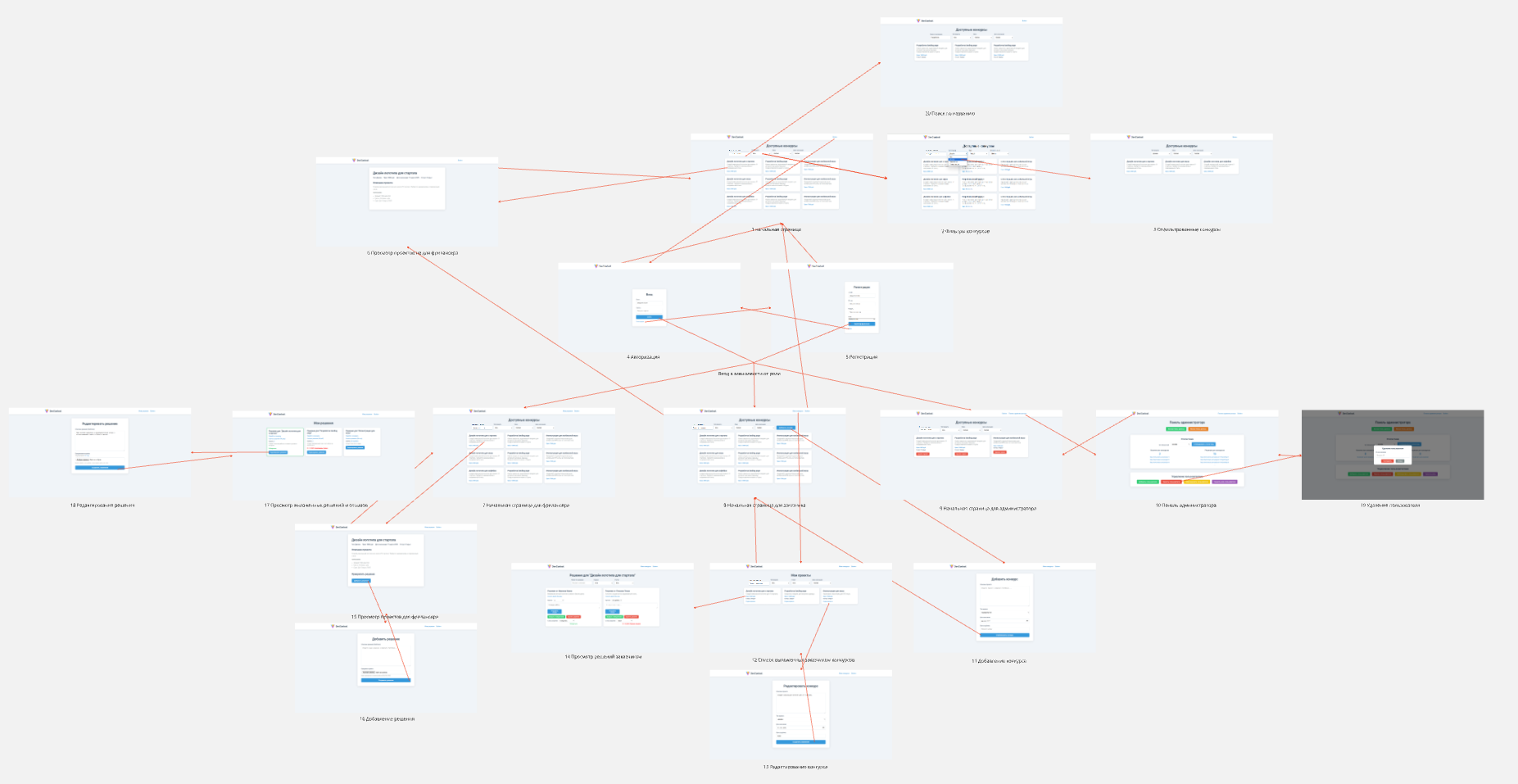
При разработке веб-приложения DevContest учитываются характеристикам системы, обеспечивающим её надёжность, удобство, масштабируемость и готовность к модернизации.

Ключевые требования:

* Надежность — система должна обеспечивать сохранность информации (профили, заявки, файлы, решения) при любых действиях, включая параллельные обращения и сбои
* Простой и интуитивно понятный интерфейс — все функции (создание конкурса, участие, подача решений, выбор победителя) должны быть логично организованы
* Прозрачная логика конкурса и активностей — пользовательский интерфейс должен обеспечивать полную информированность о состоянии конкурса, действиях участников и результатах, включая отзывы и оценки.
* Масштабируемость — система должна легко адаптироваться под рост аудитории, позволять внедрять новые функции

**2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ**

**2.1. Макет UI**

****

**2.2. Сценарии использования для задачи**

1. **Импорт данных.**
2. **Как пользователь загружает данные в программу массово (импорт из файла или внешнего источника).**

Для массового импорта/экспорта данных реализована админ-панель, с соответствующими кнопками. С помощью нее можно импортировать данные в формате zip. В импортируемом архиве хранятся данные статистики, а также JSON файл, содержащий информацию о пользователях, конкурсах, решениях и отзывах.

1. **Как пользователь загружает данные в программу вручную.**

В зависимости от роли пользователя (фрилансер, заказчик) загрузка данных на сайт отличается. В случае, если пользователь является заказчиком, используется следующий сценарий:

* Заказчик нажимает на кнопку "Добавить конкурс".
* В форме создания конкурса заказчик заполняет поля: описание проекта, тип проекта, дата окончания, приз.
* После заполнения, заказчик публикует конкурс нажав на кнопку "Опубликовать конкурс".
* Происходит возврат на начальную страницу заказчика.

В случае, если пользователь является фрилансером:

* Фрилансер выставил нужные фильтры: тип проекта, приз, дата окончания.
* Фрилансер открывает объявление нажатием на карточку.
* Фрилансер нажимает на кнопку "Добавить решение".
* Фрилансер описывает решение и прикрепляет выбранный файл, после нажимает кнопку "Отправить решение".
* Происходит возврат на начальную страницу фрилансера.

В случае, если пользователь является админом:

* Администратор просматривает объявления.
* Администратор удаляет проект нажатием на кнопку "Удалить проект".

**II.** **Представление данных.**

1. **Как данные отображаются для пользователя и что он должен сделать для их отображения (поиск, визуализация в виде графиков и таблиц)**

Данные начинают отображаться как только пользователь зашел на сайт. Все данные (конкурсы, решения, отзывы) отображаются в виде карточек, на которых добавлены соответствующие кнопки.

Для всех пользователей на карточке с конкурсом отображается:

* тип конкурса (программирование, дизайн и т.д.)
* статус конкурса (активный, завершенный)
* дата окончания
* сумма призового фонда
* описание проекта

Если пользователь является заказчиком, то на карточке с конкурсом отображаются:

* кнопка “Просмотреть решения”
* кнопка “Редактировать конкурс”

Если пользователь является фрилансером, то на карточке с конкурсом отображаются:

* кнопка “Создать решение”

Если пользователь является админом, то на карточке с конкурсом отображаются:

* кнопка “Просмотреть решения”
* кнопка “Удалить конкурс”

Для всех пользователей на карточке с решением отображаются:

* статус решения (новое, просмотрено, требуются правки, победитель, правки внесены)
* описание решения

Если пользователь является заказчиком, то на карточке с решением отображаются:

* кнопка “Просмотреть отзывы”
* кнопка “Изменить статус”
* кнопка “Оставить отзыв”

Если пользователь является фрилансером, то на карточке с конкурсом отображаются:

* кнопка “Перейти к моим решениям”
* кнопка “Просмотреть отзывы”
* кнопка “Удалить решение”
* кнопка “Редактировать решение”

Если пользователь является админом, то на карточке с конкурсом отображаются:

* кнопка “Удалить решение”

Для всех пользователей на карточке с отзывом отображаются:

* содержание отзыва

Если пользователь является заказчиком, то на карточке с решением отображаются:

* кнопка “Редактировать”
* кнопка “Удалить”

**IV. Экспорт данных.**

1. **Как пользователю получить копию хранимых в программе данных в машиночитаемом формате (JSON, XML, CSV ….)**

Для массового импорта/экспорта данных реализована админ-панель, с соответствующими кнопками. С помощью нее можно экспортировать данные о всех конкурсах, решениях, отзывах и пользователях в формате zip (статистика + JSON файл).

**2.3. Вывод о том, какие операции (чтение или запись) будут преобладать для вашего решения.**

Для данного проекта будут в большей степени преобладать операции чтения, так как большая часть пользователей, будучи фрилансерами, в первую очередь будут ознакамливаться с условиями конкурсов, прежде чем принимать участие, как и заказчики, в первую очередь будут просматривать большое количество решений, перед тем как сделать выбор в пользу одного.

**3. МОДЕЛЬ ДАННЫХ**

**3.1. Нереляционная модель данных.**

**Графическое представление модели:**

Листинг 3.1.1 – Коллекция users

| "users": {  "id": "ObjectId",  "email": "string",  "login": "string",  "password": "string",  "role": "int",  "status": "int",  "createdAt": "date"  } |
| --- |

Листинг 3.1.2 – Коллекция contests

| "contests": {  "id": "ObjectId",  "employerId": "ObjectId",  "title": "string",  "annotation": "string",  "prizepool": "int",  "description": "string",  "createdAt": "date",  "endBy": "date",  "type": "int",  "status": "int",  "winnerId": "ObjectId",  "solutions": [  {  "id": "ObjectId",  "freelancerId": "ObjectId",  "description": "string",  "status": "int",  "createdAt": "date",  "updatedAt": "date",   "reviews": [  {  "id": "ObjectId",  "score": "float",  "commentary": "string",  "createdAt": "date",  "updatedAt": "date"  }  ]  }  ]  } |
| --- |

**Описание назначений коллекций, типов данных и сущностей:**

Таблица 3.1.1 – Коллекция users (Пользователи)

| **Поле** | **Тип** | **Размер (байт)** | **Описание** |
| --- | --- | --- | --- |
| id | ObjectId | 12B | Уникальный идентификатор пользователя |
| email | string | 60B | Почта пользователя |
| login | string | 20B | Логин пользователя |
| password | string | 60B | Хеш пароля пользователя |
| role | int | 4B | Роль (1 - заказчик, 2 - фрилансер, 3 - админ) |
| status | int | 4B | Статус (1 - активный, 2 - заблокированный) |
| createdAt | date | 8B | Дата создания аккаунта |
| id | ObjectId | 12B | Уникальный идентификатор пользователя |

Таблица 3.1.2 – Коллекция contests (Конкурсы)

| **Поле** | **Тип** | **Размер (байт)** | **Описание** |
| --- | --- | --- | --- |
| id | ObjectId | 12B | Уникальный идентификатор конкурса |
| employerId | ObjectId | 12B | ID заказчика (ссылка на users) |
| title | string | 50B | Название конкурса |
| annotation | string | 150B | Краткое описание конкурса |
| prizepool | int | 4B | Призовой фонд конкурса |
| description | string | 2000B | Полное описание конкурса |
| createdAt | date | 8B | Дата создания конкурса |
| endBy | date | 8B | Дата окончания конкурса |
| type | int | 4B | Тип конкурса (1 - дизайн, 2 - разработка, 3 - иллюстрация, …) |
| status | int | 4B | Статус конкурса (1 - активный, 2 - завершённый, 3 - отменённый, на проверке) |
| winnerId | ObjectId | 12B | ID победителя (ссылка на users) |

Таблица 3.1.3 – Вложенный документ solutions (Решения конкурса; вложен в contests)

| **Поле** | **Тип** | **Размер (байт)** | **Описание** |
| --- | --- | --- | --- |
| id | ObjectId | 12B | Уникальный идентификатор решения |
| freelancerId | ObjectId | 12B | ID фрилансера (ссылка на users) |
| description | string | 100B | Описание решения |
| status | int | 4B | Статус решения (1 - новое, 2 - победитель, 3 - необходимы правки, 4 - правки внесены) |
| createdAt | date | 8B | Дата загрузки решения |
| updatedAt | date | 8B | Дата обновления решения |

Таблица 3.1.4 – Вложенный документ reviews (Отзывы на решение; вложен в solutions)

| **Поле** | **Тип** | **Размер (байт)** | **Описание** |
| --- | --- | --- | --- |
| id | ObjectId | 12B | Уникальный идентификатор отзыва |
| score | float | 4B | Оценка решения (0.0 – 10.0) |
| commentary | string | 100B | Комментарий к решению |
| createdAt | date | 8B | Дата создания отзыва |
| updatedAt | date | 8B | Дата обновления отзыва |

**Оценка объема информации, хранимой в модели:**

### Пользователи (*users*)

Каждый пользователь занимает:

id (12B) + email (60B) + login (20B) + password (60B) + role (4B) + status (4B) + createdAt (8B) = **168B**.

Если в системе U пользователей, то общий объем данных для пользователей: U \* 168.

### Конкурсы (*contests*)

Каждый конкурс занимает:

id (12B) + employerId (12B) + title (50B) + annotation (150B) + prizepool (4B) + description (2000B) + createdAt (8B) + endBy (8B) + type (4B) + status (4B) + winnerId (12B) = **2264B**.

Если в системе C конкурсов, то общий объем данных: C \* 2264.

### Решения (*solutions*)

Каждое решение занимает:

id (12B) + freelancerId (12B) + description (100B) + status (4B) + createdAt (8B) + updatedAt (8B) = **144B**.

Пусть в среднем на один конкурс приходится S решений, тогда общий объем данных: C \* S \* 144.

### Отзывы (*reviews*)

Каждый отзыв занимает:

id (12B) + score (4B) + commentary (100B) + createdAt (8B) + updatedAt (8B) = **132B**.

Пусть в среднем на одно решение приходится R отзывов, тогда общий объем данных: C \* S \* R \* 132.

### Формула (зависимость от U, C, S, R)

Общий объем хранимых данных:

U \* 168 + C \* (2264 + S \* (144 + R \* 132)),

где U - количество пользователей, C - количество конкурсов, S - среднее число решений на конкурс, R - среднее число отзывов на решение.

### Итоговая формула

Пусть в среднем на 1 заказчика приходится 1000 фрилансеров. Пусть 1 фрилансер в среднем отправляет 3 решения и на каждое решение приходится по 1 отзыву. Тогда, U = 1000 + 1 = 1001, C = 1, S = 1000 \* 3 = 3000, R = 1.

Пусть N - количество пользователей, тогда:

((U \* 168 + C \* (2264 + S \* (144 + R \* 132))) / (U)) \* N = (1001 \* 168 + 1 \* (2264 + 3000 \* (144 + 1 \* 132))) / (1001) \* N = 998 \* N.

Формула: 998 \* N.

## 

## Примеры запросов:

### Сценарий: “Главная страница”

Листинг 3.1.3 – Фильтрация конкурсов по параметрам

| db.contests.find({  "type": 1,  "prizepool": { "$gte": 1000 },  "endBy": { "$gte": new Date() }  }).sort({ "createdAt": -1 }); |
| --- |

Листинг 3.1.4 – Поиск по названию конкурса

| db.contests.find({ "title": "дизайн" }); |
| --- |

Листинг 3.1.5 – Просмотр деталей конкурса

| db.contests.findOne({ "\_id": ObjectId("661a2b3c4d5e6f001d8e9012") }); |
| --- |

Число запросов: 1-3 (в зависимости от действий).

Коллекции: *contests*.

### Сценарий: “Вход пользователя”

Листинг 3.1.6 – Вход пользователя

| db.users.findOne({ "login": "freelancer123", "password": "hashed\_password" }); |
| --- |

Число запросов: 1.

Коллекции: *users*.

### Сценарий: “Регистрация пользователя”

Листинг 3.1.7 – Регистрация пользователя

| db.users.insertOne({  "email": "newuser@example.com",  "login": "new\_user",  "password": "hashed\_password",  "role": 2,  "status": 1,  "createdAt": new Date()  }); |
| --- |

Число запросов: 1.

Коллекции: *users*.

### Сценарий: “Начальная страница фрилансера”

Листинг 3.1.8 – Поиск доступных конкурсов

| db.contests.find({ "status": 1 }).sort({ "createdAt": -1 }); |
| --- |

Листинг 3.1.9 – Добавление решения к конкурсу

| db.contests.updateOne(  { "\_id": ObjectId("661a2b3c4d5e6f001d8e9012") },  { "$push": { "solutions": {  "id": ObjectId(),  "freelancerId": ObjectId("660a1f9e5f1b2c001c8a1234"),  "description": "Новое решение",  "status": 1,  "createdAt": new Date(),  "updatedAt": new Date(),  "reviews": []  } } }  ); |
| --- |

Число запросов: 1-2.

Коллекции: *contests*.

### Сценарий: Мои решения

Листинг 3.1.10 – Просмотр своих решений

| db.contests.find({ "solutions.freelancerId": ObjectId("660a1f9e5f1b2c001c8a1234") },  { "solutions.$": 1 }); |
| --- |

Листинг 3.1.11 – Редактирование решения

| db.contests.updateOne(  { "solutions.id": ObjectId("661a3d4e5f6g7h001e9f0123") },  { "$set": { "solutions.$.description": "Обновленное решение", "solutions.$.updatedAt": new Date() } }  ); |
| --- |

#### 

Число запросов: 1-2.

Коллекции: *contests*.

### Сценарий: Начальная страница заказчика

Листинг 3.1.12 – Добавление конкурса

| db.contests.insertOne({  "employerId": ObjectId("660a1f9e5f1b2c001c8a5678"),  "title": "Новый конкурс",  "annotation": "Краткое описание",  "prizepool": 3000,  "description": "Полное описание",  "createdAt": new Date(),  "endBy": new Date("2025-04-01"),  "type": 2,  "status": 1,  "winnerId": null,  "solutions": []  }); |
| --- |

Число запросов: 1.

Коллекции: *contests*.

### Сценарий: Мои конкурсы

Листинг 3.1.13 – Просмотр конкурсов заказчика

| db.contests.find({ "employerId": ObjectId("660a1f9e5f1b2c001c8a5678") }); |
| --- |

#### 

Листинг 3.1.14 – Поиск решений по статусу и оценке

| db.contests.find({ "solutions.status": 1, "solutions.reviews.score": { "$gte": 8.0 } }); |
| --- |

#### 

Листинг 3.1.15 – Выбор победителя

| db.contests.updateOne(  { "\_id": ObjectId("661a2b3c4d5e6f001d8e9012") },  { "$set": { "winnerId": ObjectId("660a1f9e5f1b2c001c8a1234"), "status": 2 } }  ); |
| --- |

Число запросов: 1-3.

Коллекции: *contests*.

### Сценарий: Начальная страница администратора

Листинг 3.1.16 – Удаление конкурса

| db.contests.deleteOne({ "\_id": ObjectId("661a2b3c4d5e6f001d8e9012") }); |
| --- |

Число запросов: 1-3.

Коллекции: *contests*.

### Сценарий: Панель администратора

Листинг 3.1.17 – Блокировка пользователя

| db.users.updateOne(  { "\_id": ObjectId("660a1f9e5f1b2c001c8a1234") },  { "$set": { "status": 2 } }  ); |
| --- |

Число запросов: 1.

Коллекции: *users*.

**3.2. Реляционная модель данных.**

**Графическое представление модели:**

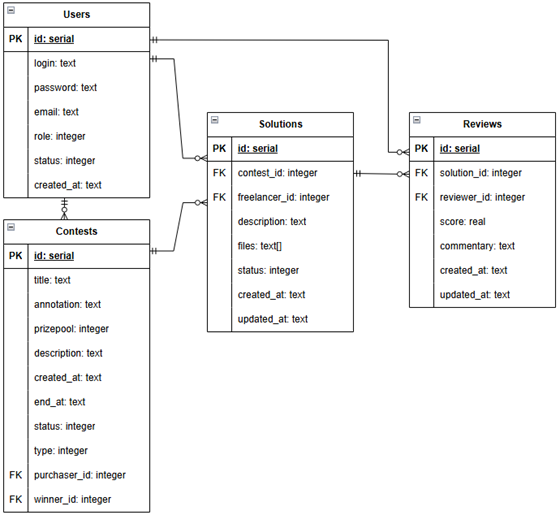
****

Рисунок 3.2.1 – Графическое представление модели

**Описание назначений коллекций, типов данных и сущностей:**

Users - таблица пользователей: - id - идентификационный номер пользователя - login - логин пользователя - email - почта пользователя - password - пароль пользователя - role - роль пользователя (заказчик, фрилансер, администратор) - status - статус пользователя (активный, заблокированный) - created\_at - дата создания аккаунта.

Contests - таблица конкурсов: - id - идентификационный номер конкурса - title - название конкурса - annotation - краткое описание конкурса - prizepool - призовой фонд конкурса - description - полное описание конкурса - created\_at - дата создания конкурса - end\_at - дата окончания конкурса - status - статус конкурса (активный, завершенный, отмененный, на проверке) - type - тип конкурса - purchased\_id - идентификационный номер заказчика - winner\_id - идентификационный номер победителя.

Solutions - таблица отправленных решений: - id - идентификационный номер решения - contest\_id - идентификационный номер конкурса - freelancer\_id - идентификационный номер фрилансера - description - текст решения - files - пути к загруженным файлам - status - статус решения - created\_at - дата создания решения - updated\_at - дата обновления решения.

Reviews - таблица отзывов на решения: - id - идентификационный номер отзыва - solution\_id - идентификационный номер решения - reviewer\_id - идентификационный номер заказчика (кем был оставлен отзыв) - score - оценка решения (от 0.0 до 10.0) - commentary - комментарий к решению - created\_at - дата создания отзыва - updated\_at - дата обновления отзыва.

**Оценка объема информации, хранимой в модели:**

*Users*: id - 4 байта, login - 20 байт, password - 60 байт, email - 60 байт, role - 4 байта, status - 4 байта, created\_at - 20 байт. Суммарно - 172 байт.  
 *Contests*: три id - 12 байт, title - 50 байт, annotation - 150 байт, prizepool - 4 байта, description - 2000 байт, две даты - 40 байт, status - 4 байта, type - 4 байта. Суммарно - 2264 байт.  
 *Solutions*: три id - 12 байт, description - 100 байт, status - 4 байта, files - 50 байт, две даты - 40 байт. Суммарно - 206 байт.  
 *Reviews*: три id - 12 байт, score - 4 байта, commentary - 100 байт, две даты - 40 байт. Суммарно - 156 байт.

При регистрации пользователя создается запись в таблице Users, которая занимает 172 байт. В среднем на 1 заказчика приходится 1000 фрилансеров. Если фрилансер, то в среднем он отправляет 3 решения, что занимает 618 байт. На решение в среднем приходит 1 отзыв, значит будет 3 отзыва, что занимает 468 байт. Следовательно, на одного фрилансера приходится 1258 байт. Если же заказчик, то на него приходится в среднем 1 конкурс, что занимает 2264 байт, следовательно на одного заказчика приходится 2436 байт. Отзывы заказчика уже включены.  
 Получаем следующую формулу: N \* 1260.

**Примеры запросов:**

**Сценарий использования “Главная страница”:**

Листинг 3.2.1 – Основной сценарий: Поиск конкурса по типу конкурса, призу, времени окончания

| SELECT \* FROM Contests  WHERE type=”Разработка игр” AND prizepool=5000 AND end\_at=2025-03-21T12:30:25 |
| --- |

Листинг 3.2.2 – Для отображения страницы конкурса требуется вся информация о нем

| SELECT \* FROM Contests  WHERE id=3 |
| --- |

Листинг 3.2.3 – Альтернативный сценарий

| SELECT \* FROM Contests  WHERE title LIKE “%Game%” |
| --- |

Листинг 3.2.4 – Для отображения страницы конкурса требуется вся информация

| SELECT \* FROM Contests  WHERE id=3 |
| --- |

**Сценарий использования “Вход пользователя”:**

Листинг 3.2.5 – Основной сценарий: Проверка данных при входе

| SELECT role FROM Users  WHERE login = 'existing\_user’ AND password = SHA2('secret\_password', 256)  AND status = 0; |
| --- |

Альтернативный сценарий - “Регистрация пользователя”\

**Сценарий использования “Регистрация пользователя”:**

Листинг 3.2.6 – Основной сценарий: Пользователь регистрируется, заполняя форму

| INSERT INTO Users (login, email, password, role, status, created\_at)  VALUES ('new\_freelancer', 'user@example.com', SHA2('my\_password', 256),   ‘freelancer’, 0, NOW()); |
| --- |

Листинг 3.2.7 – Получение роли нового пользователя для отображения начальной страницы

| SELECT role FROM Users  WHERE login = 'new\_freelancer’ |
| --- |

Альтернативный сценарий - “Вход пользователя”

**Сценарий использования “Начальная страница фрилансера”:**

Листинг 3.2.8 – Основной сценарий: Поиск конкурса по типу конкурса, призу, времени окончания

| SELECT \* FROM Contests  WHERE type=”Разработка игр” AND prizepool=5000 AND end\_at=2025-03-21T12:30:25 |
| --- |

Листинг 3.2.9 – Для отображения страницы конкурса требуется вся информация

| SELECT \* FROM Contests  WHERE id=3 |
| --- |

Листинг 3.2.10 – Добавление решения

| INSERT INTO Solutions (contest\_id, freelancer\_id, description, status, created\_at, updated\_at)  VALUES (15, 5, 'Моё решение /uploads/solution\_freelancer5.zip', 2, NOW(), NOW()); |
| --- |

Листинг 3.2.11 – Альтернативный сценарий: Поиск конкурса по названию

| SELECT \* FROM Contests  WHERE title LIKE “%Design%” |
| --- |

Листинг 3.2.12 – Для отображения страницы конкурса требуется вся информация

| SELECT \* FROM Contests  WHERE id=5 |
| --- |

**Сценарий использования “Мои решения”:**

Листинг 3.2.13 – Основной сценарий: Получение всех решений фрилансера

| SELECT Solutions.\*, Contests.title AS contest\_title, Contests.end\_at, Reviews.score FROM Solutions  JOIN Contests ON Solutions.contest\_id = Contests.id  LEFT JOIN Reviews ON Solutions.id = Reviews.solution\_id  WHERE Solutions.freelancer\_id = 5 AND Solutions.id = 9; |
| --- |

Листинг 3.2.14 – Переход к редактированию решения

| SELECT S.\*, R.\*  FROM Solutions AS S  LEFT JOIN Reviews AS R ON S.id = R.solution\_id  WHERE S.contest\_id = 4 AND S.freelancer\_id = 5; |
| --- |

Листинг 3.2.15 – Редактирование решения

| INSERT INTO Solutions (contest\_id, freelancer\_id, description, status, updated\_at)  VALUES (4, 5, 'Исправленное решение /uploads/solution\_freelancer6.zip', 2, NOW()); |
| --- |

Листинг 3.2.16 – Альтернативный сценарий: Получение всех решений фрилансера

| SELECT Solutions.\*, Contests.title AS contest\_title, Contests.end\_at, Reviews.score FROM Solutions  JOIN Contests ON Solutions.contest\_id = Contests.id  LEFT JOIN Reviews ON Solutions.id = Reviews.solution\_id  WHERE Solutions.freelancer\_id = 5; |
| --- |

Листинг 3.2.17 – Открытие решений конкретного конкурса

| SELECT S.\*, R.\*  FROM Solutions AS S  LEFT JOIN Reviews AS R ON S.id = R.solution\_id  WHERE S.contest\_id = 4 AND S.freelancer\_id = 5; |
| --- |

**Сценарий использования “Начальная страница заказчика”:**

Листинг 3.2.18 – Основной сценарий: Создание нового конкурса

| INSERT INTO Contests (annotation, prizepool, description, created\_at, end\_at, status, type, purchaser\_id, winner\_id)  VALUES (‘Разработать макет сайта', 5000, ‘Нужно разработать макет сайта для кафе’   NOW(), ‘2025-06-11T23:59:59’, 0, 4, 8, null); |
| --- |

Листинг 3.2.19 – Возврат на начальную страницу заказчика

| SELECT \* FROM Contests  WHERE purchaser\_id = 8  ORDER BY created\_at DESC; |
| --- |

Листинг 3.2.20 – Альтернативный сценарий: Переход на страницу конкурсов заказчика

| SELECT \* FROM Contests  WHERE purchaser\_id = 8  ORDER BY created\_at DESC; |
| --- |

**Сценарий использования “Мои конкурсы”:**

Листинг 3.2.21 – Основной сценарий: Открытие страницы конкурсов заказчика

| SELECT \* FROM Contests  WHERE purchaser\_id = 8  ORDER BY created\_at DESC; |
| --- |

Листинг 3.2.22 – Заказчик нажимает на карточку конкурса и переходит к просмотру загруженных решений

| SELECT S.id AS solution\_id, S.freelancer\_id, S.description,  S.status, R.score, R.commentary, U.login  FROM Solutions AS S  JOIN Users AS U ON S.freelancer\_id = U.id  LEFT JOIN Reviews AS R ON S.id = R.solution\_id  WHERE S.contest\_id = 10; |
| --- |

Листинг 3.2.23 – Заказчик выбирает конкретное решение

| SELECT S.id AS solution\_id, S.freelancer\_id, S.description,  S.file\_url, S.status, R.score, R.commentary, U.login  FROM Solutions AS S  JOIN Users AS U ON S.freelancer\_id = U.id  LEFT JOIN Reviews AS R ON S.id = R.solution\_id  WHERE S.id = 25; |
| --- |

Листинг 3.2.24 – Альтернативный сценарий: Получение информации о конкретном конкурсе заказчика

| SELECT \* FROM Contests  WHERE id = 10 AND purchaser\_id = 7; |
| --- |

Листинг 3.2.25 – Редактирование конкурса

| UPDATE Contests  SET description = 'Изменённое описание конкурса'  WHERE id = 10 AND purchaser\_id = 7; |
| --- |

**Сценарий использования “Начальная страница администратора”:**

Листинг 3.2.26 – Основной сценарий: Администратор просматривает объявления

| SELECT \* FROM Contests  WHERE status = 3; |
| --- |

Листинг 3.2.27 – Если конкурс некорректен, то администратор удаляет конкурс

| DELETE FROM Contests  WHERE id = 15 AND status = 3; |
| --- |

Листинг 3.2.28 – Если конкурс корректный, то администратор одобряет конкурс

| UPDATE Contests  SET status = 'active'  WHERE id = 15 AND status = 3; |
| --- |

**Сценарий использования “Панель администратора”:**

Листинг 3.2.29 – Основной сценарий: Экспорт данных

| SELECT \*  INTO OUTFILE '/var/lib/mysql-files/contests\_export.csv'  FIELDS TERMINATED BY ','  ENCLOSED BY '"'  LINES TERMINATED BY '\n'  FROM Contests; |
| --- |

Листинг 3.2.30 – Импорт данных

| LOAD DATA INFILE '/var/lib/mysql-files/contests\_export.csv'  INTO TABLE Contests  FIELDS TERMINATED BY ','  ENCLOSED BY '"'  LINES TERMINATED BY '\n'; |
| --- |

Листинг 3.2.31 – Альтернативный сценарий: Просмотр статистики по конкурсам

| SELECT  COUNT(\*) AS total\_contests,  SUM(prizepool) AS total\_prizepool,  AVG(prizepool) AS avg\_prizepool  FROM Contests; |
| --- |

Листинг 3.2.31 – Блокировка пользователя

| UPDATE Users  SET status = 3  WHERE id = 10; |
| --- |

Листинг 3.2.32 – Смена роли пользователя

| UPDATE Users  SET role = 0  WHERE id = 11; |
| --- |

Количество запросов для совершения юзкейсов

* Открытие конкурса: 2 запроса
* Вход пользователя: 1 запрос
* Регистрация пользователя: 2 запроса
* Начальная страница фрилансера: 2/3 запроса
* Мои решения: 2/3 запроса
* Начальная страница заказчика: 1/2 запроса
* Мои конкурсы: 2/3 запроса
* Начальная страница администратора: 2 запроса
* Панель администратора: 1/2 запроса

**Количество задействованный коллекций:**

Четыре сущности: Users, Contests, Reviews, Solutions

**3.3. Сравнение моделей.**

Реляционная модель требует больше памяти, чем нереляционная (SQL: 1260N vs NoSQL: 998N для N – общего количества пользователей), хоть избыточность нереляционной модели и немного выше (NoSQL: 1.037 vs SQL: 1.019).

Количество запросов для выполнения юзкейсов в моделях примерно одинаковое, так как в обеих моделях каждое действие юзкейса требует в среднем один запрос.

Реляционная модель задействует большее количество сущностей, чем нереляционная коллекций (SQL: 4 vs NoSQL: 2).

## Вывод:

Реляционная модель проигрывает нереляционной по объёму занимаемой памяти. В количестве запросов модели схожи.

**4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ**

**4.1. Краткое описание (из каких модулей / контейнеров состоит, какую архитектуру вы использовали)**

Приложение представляет собой веб-платформу для организации конкурсов, где заказчики публикуют задания, фрилансеры предлагают решения, а администраторы управляют типами конкурсов и статистикой, а также осуществляют модерацию. Оно состоит из двух основных модулей и базы данных, развернутых в контейнерах с использованием Docker:

* Фронтенд: Реализован на React с использованием React Bootstrap. Отвечает за пользовательский интерфейс: отображение конкурсов, формы для создания/редактирования конкурсов, панель администратора для управления типами конкурсов, импорта/экспорта данных и просмотра статистики. Развернут как контейнер web, доступный на порту 3000.
* Бэкенд: Реализован на Flask, предоставляет REST API для обработки запросов на создание, обновление, фильтрацию и получение данных о конкурсах, решениях, отзывах и пользователях. Взаимодействует с MongoDB. Развернут как контейнер server, доступный на порту 8000.
* База данных (контейнер db): MongoDB, развернутая как контейнер db, хранит данные о пользователях, конкурсах, решениях и отзывах в виде документов, обеспечивая гибкость и масштабируемость

Коллекции:

* + users: email, login, password, role (1 — фрилансер, 2 — заказчик, 3 — админ), status.
  + contests: title, annotation, description, prizepool, employerId (ссылка на users.\_id), endBy, type (ссылка на contest\_types.\_id), status, winnerId.
  + solutions: contestId, freelancerId, title, annotation, description, files, status, reviews (вложенные документы).
  + reviews: reviewerId, score, commentary.
  + contest\_types: name.

Архитектура — клиент-серверная, с контейнеризацией на основе Docker Compose:

* Клиентская часть (React, контейнер web) рендерит интерфейс, валидирует данные через MobX и отправляет запросы к серверу с помощью Axios.
* Серверная часть (Flask, контейнер server) обрабатывает запросы, валидирует данные с Pydantic, управляет файлами и взаимодействует с MongoDB.
* MongoDB (контейнер db) обеспечивает хранение данных, с проверкой работоспособности через healthcheck.
* Docker Compose управляет зависимостями между сервисами: фронтенд зависит от бэкенда, бэкенд — от базы данных, с настройкой переменных окружения (например, MONGO\_URI, DB\_NAME) для связи.

**4.2. Использованные технологии**

Фронтенд:

* React — для создания интерфейса.
* React Bootstrap — для стилизации и адаптивного дизайна.
* MobX — для управления состоянием.
* Axios — для HTTP-запросов.

Бэкенд:

* Flask — для REST API.
* Pydantic — для валидации и сериализации данных.
* MongoDB — NoSQL база данных.
* Werkzeug — для безопасной обработки файлов.

Контейнеризация:

* Docker — для создания контейнеров фронтенда (web), бэкенда (server) и базы данных (db).
* Docker Compose — для управления зависимостями и настройки портов.

**4.3. Снимки экрана приложения**

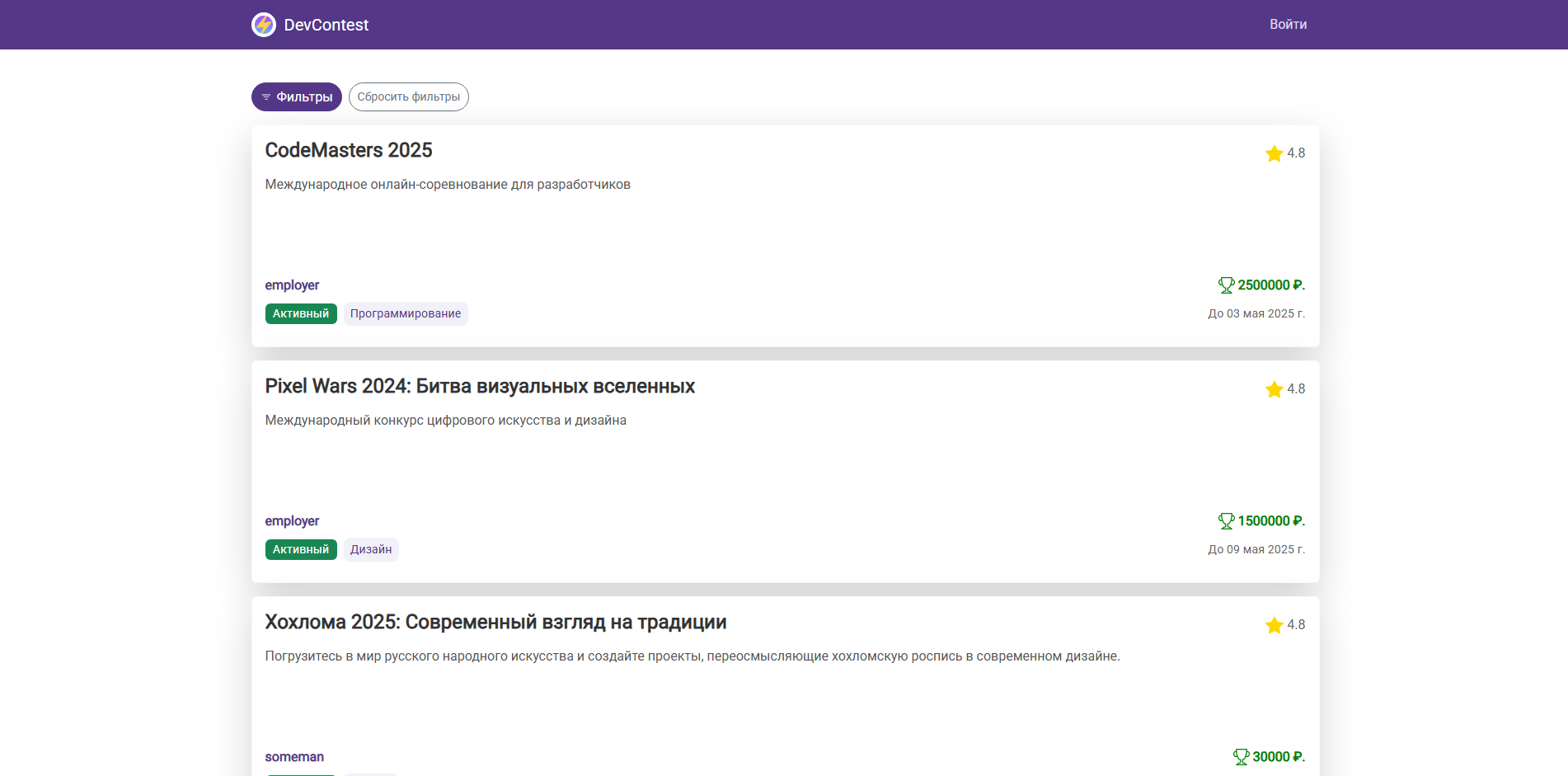
****

Рисунок 1 - Главная страница

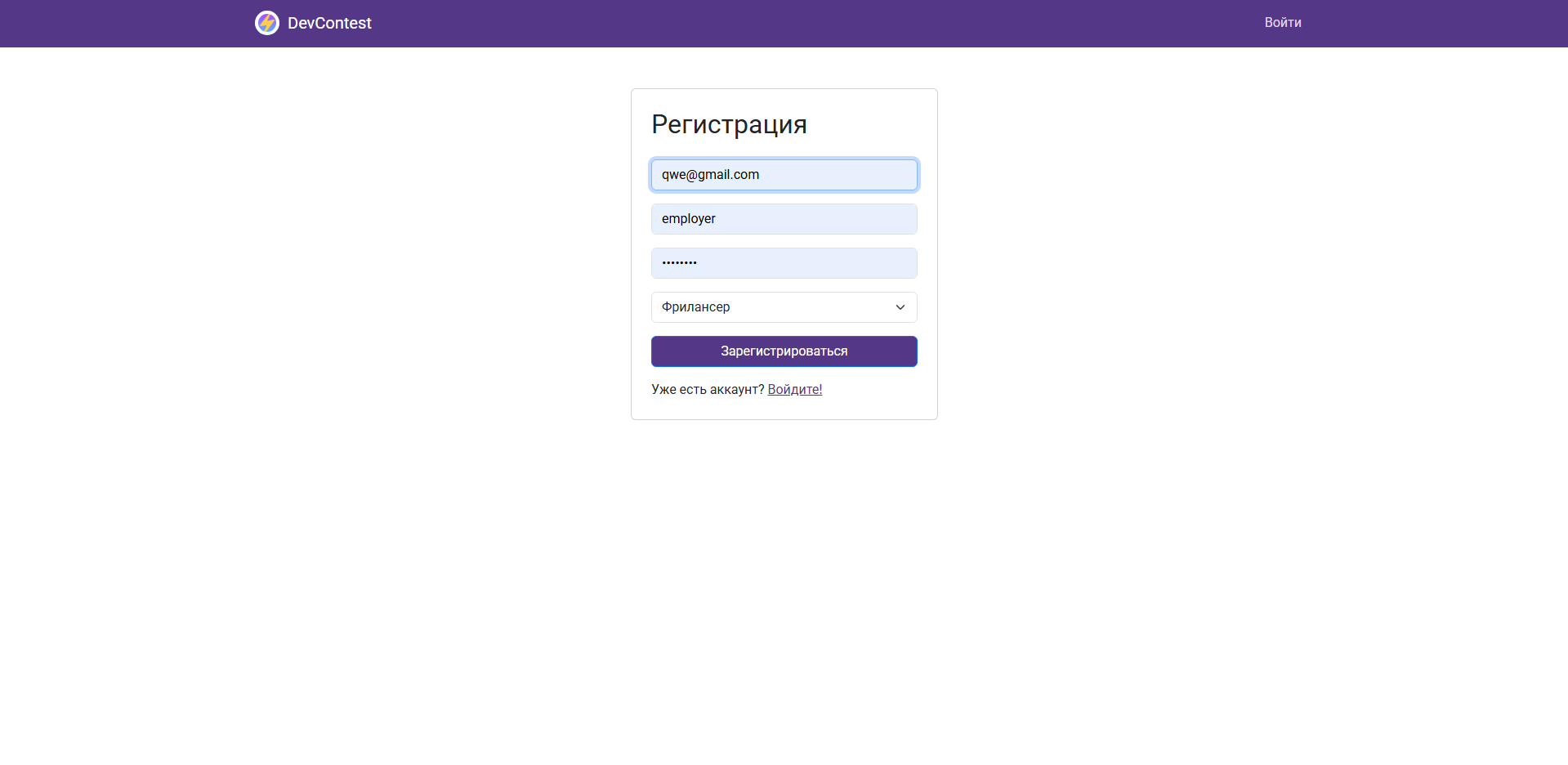


Рисунок 2 - Страница регистрации

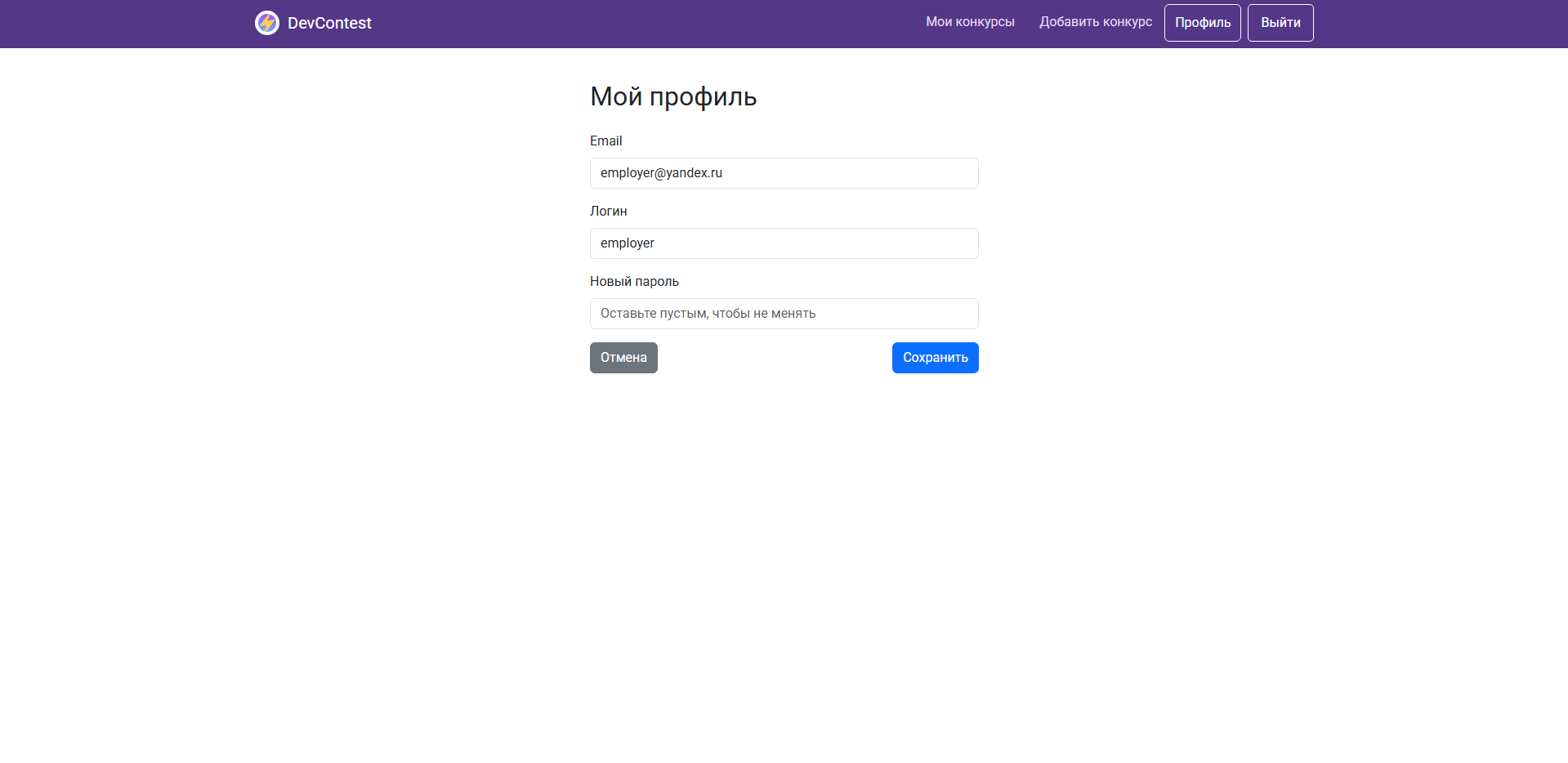


Рисунок 3 - Страница профиля

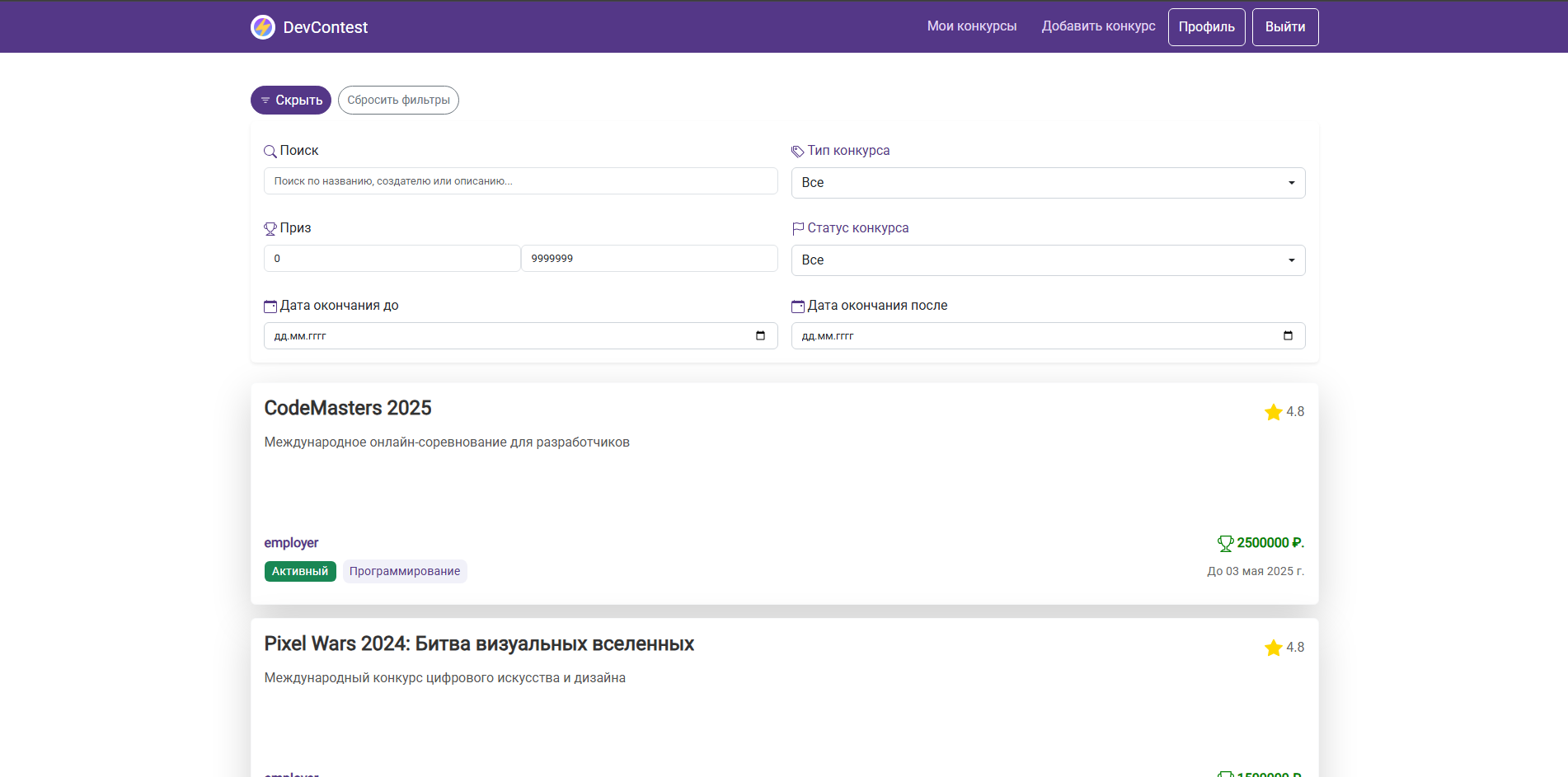


Рисунок 4 - Фильтры на главной странице

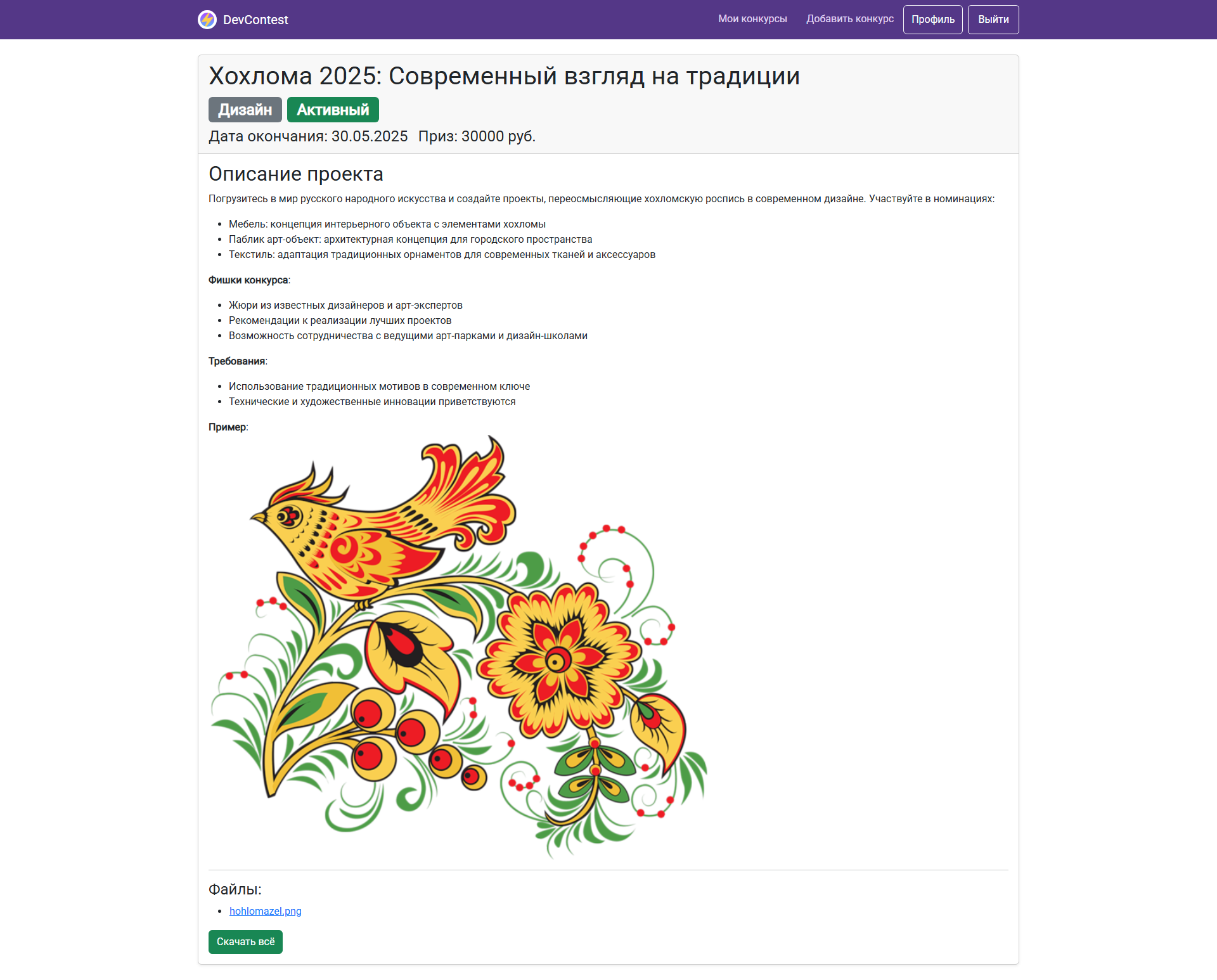


Рисунок 5 - Страница конкурса

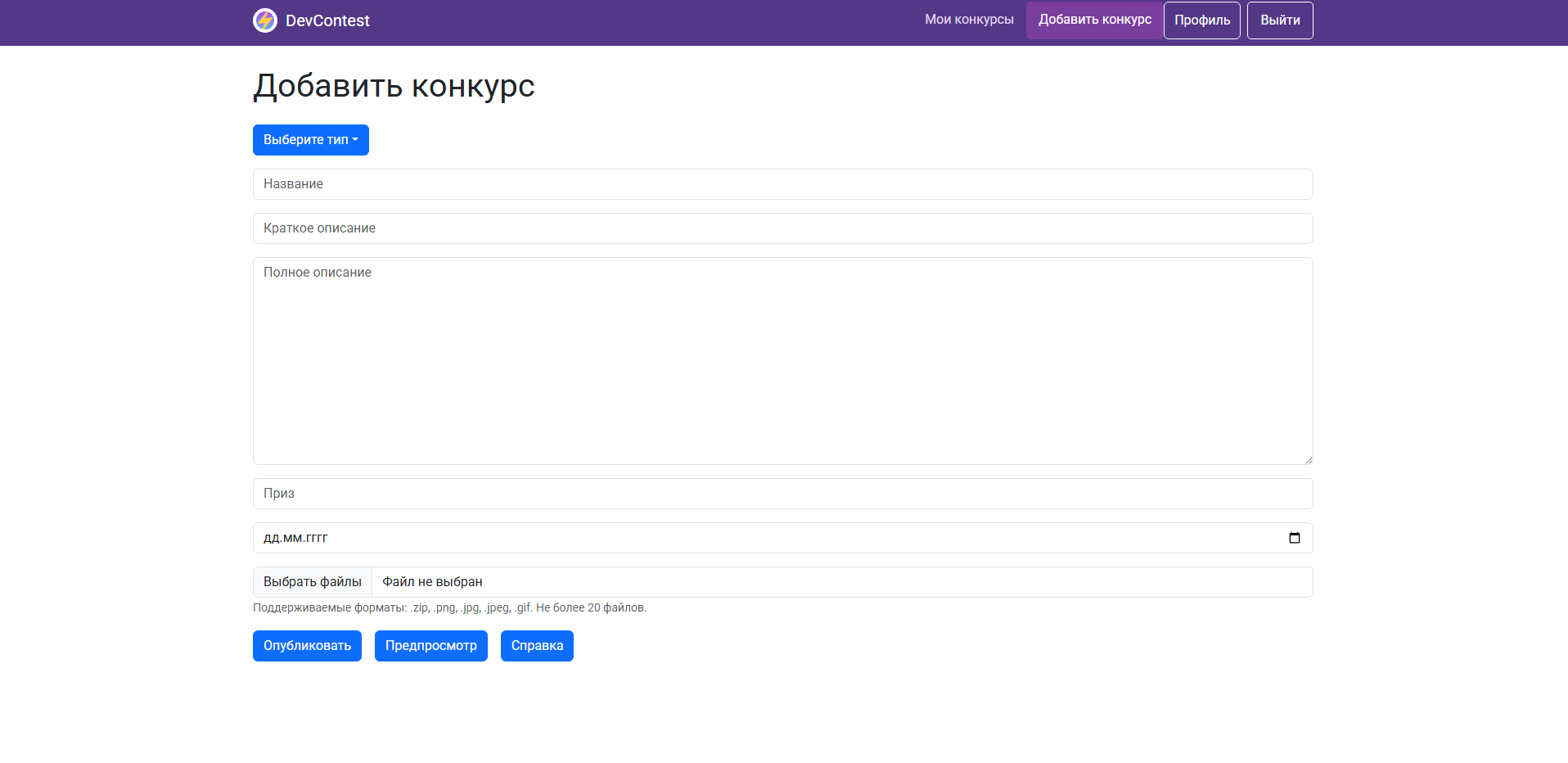


Рисунок 6 - Страница создания конкурса

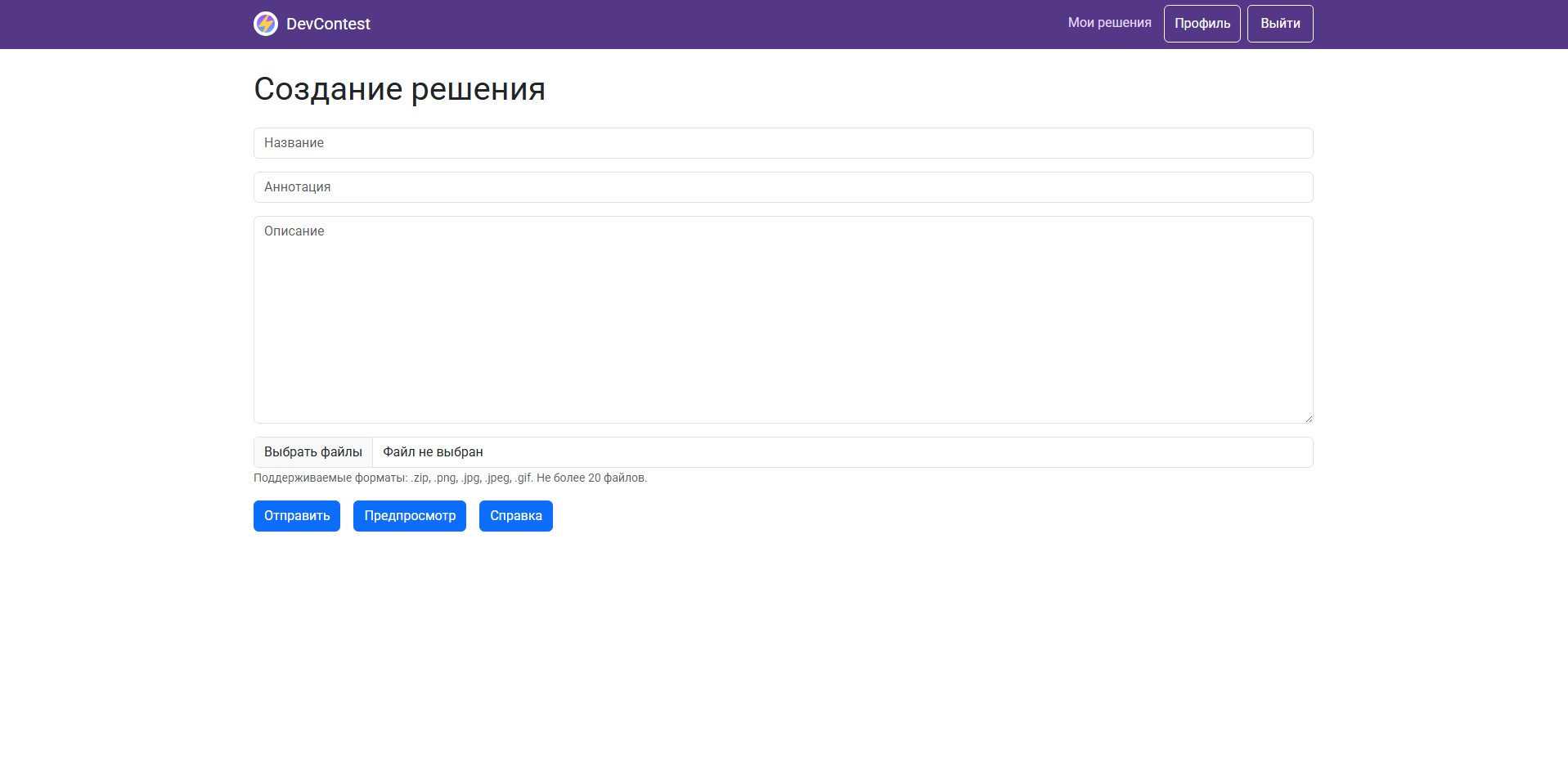


Рисунок 7 - Страница создания решения

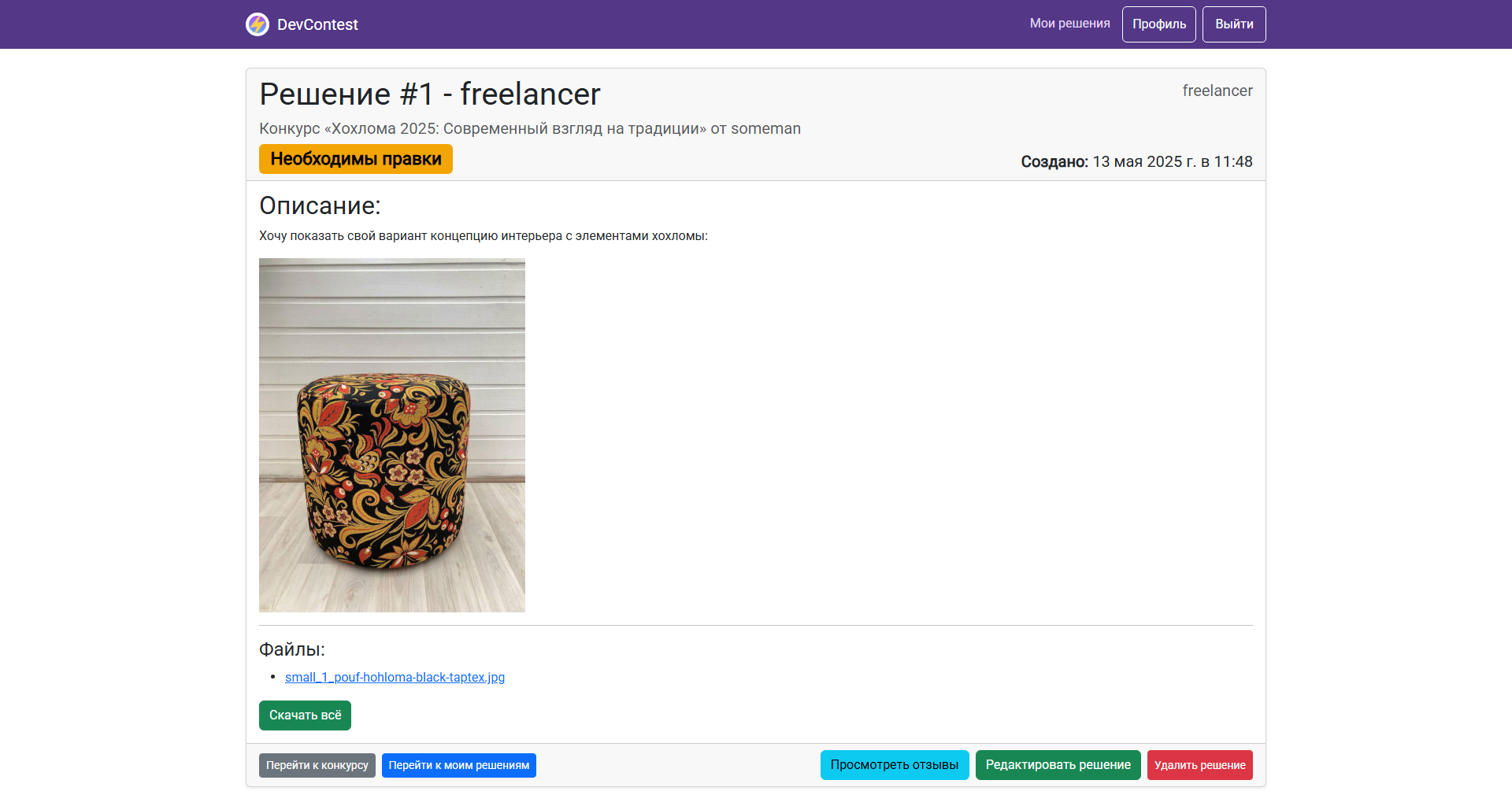


Рисунок 8 - Страница решения

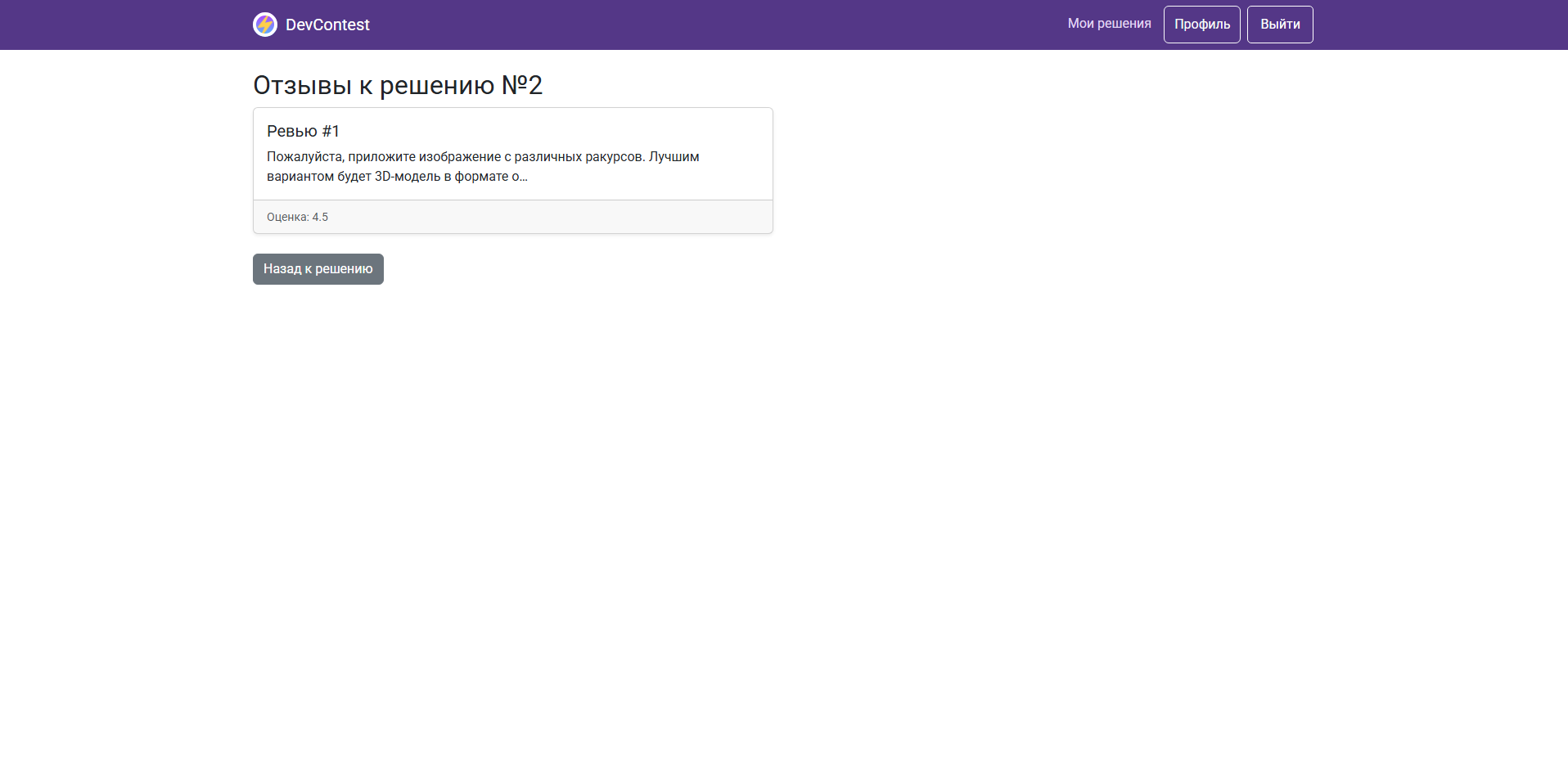


Рисунок 9 - Страница с отзывами на решение

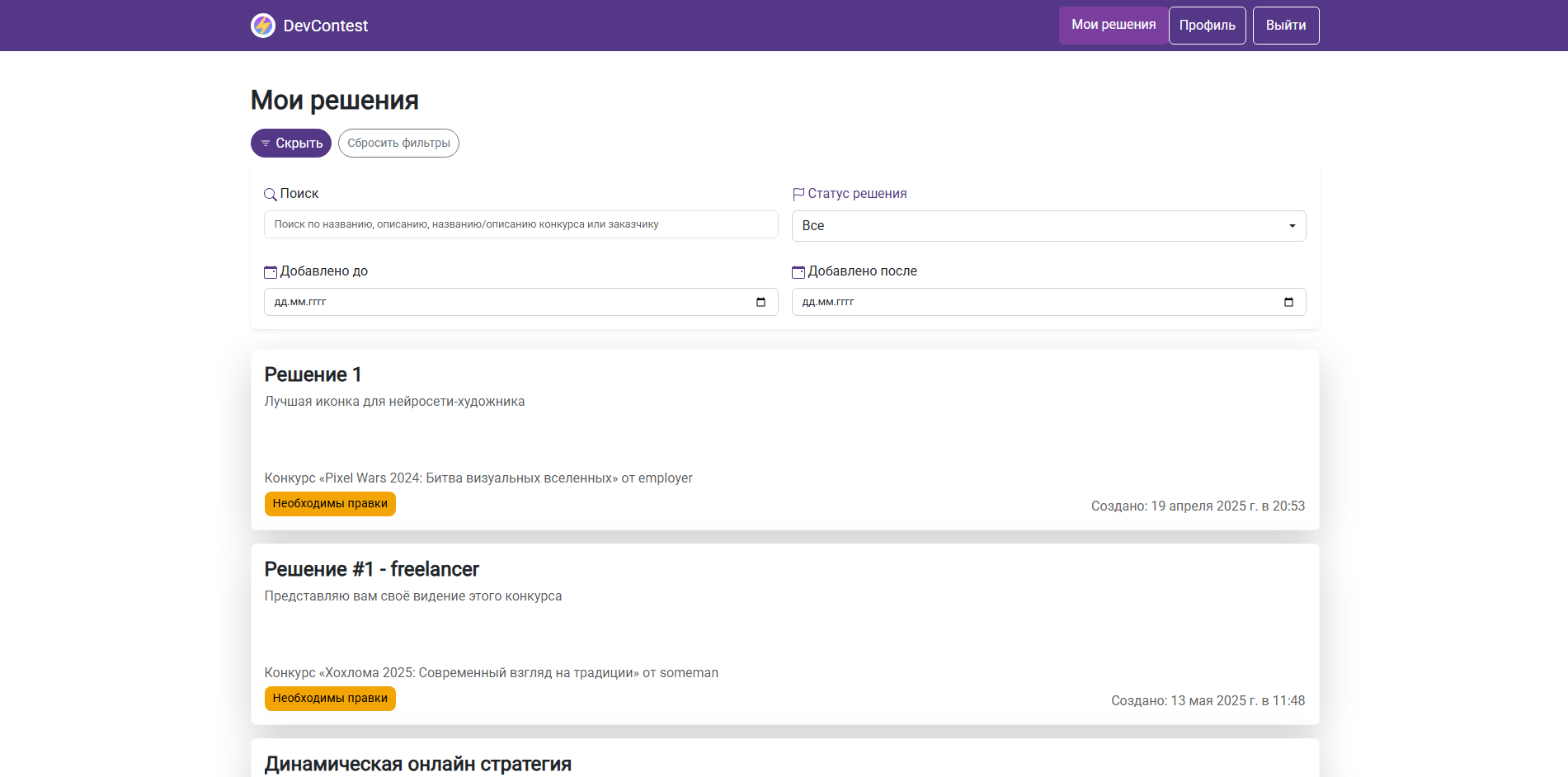


Рисунок 10 - Фильтры решений

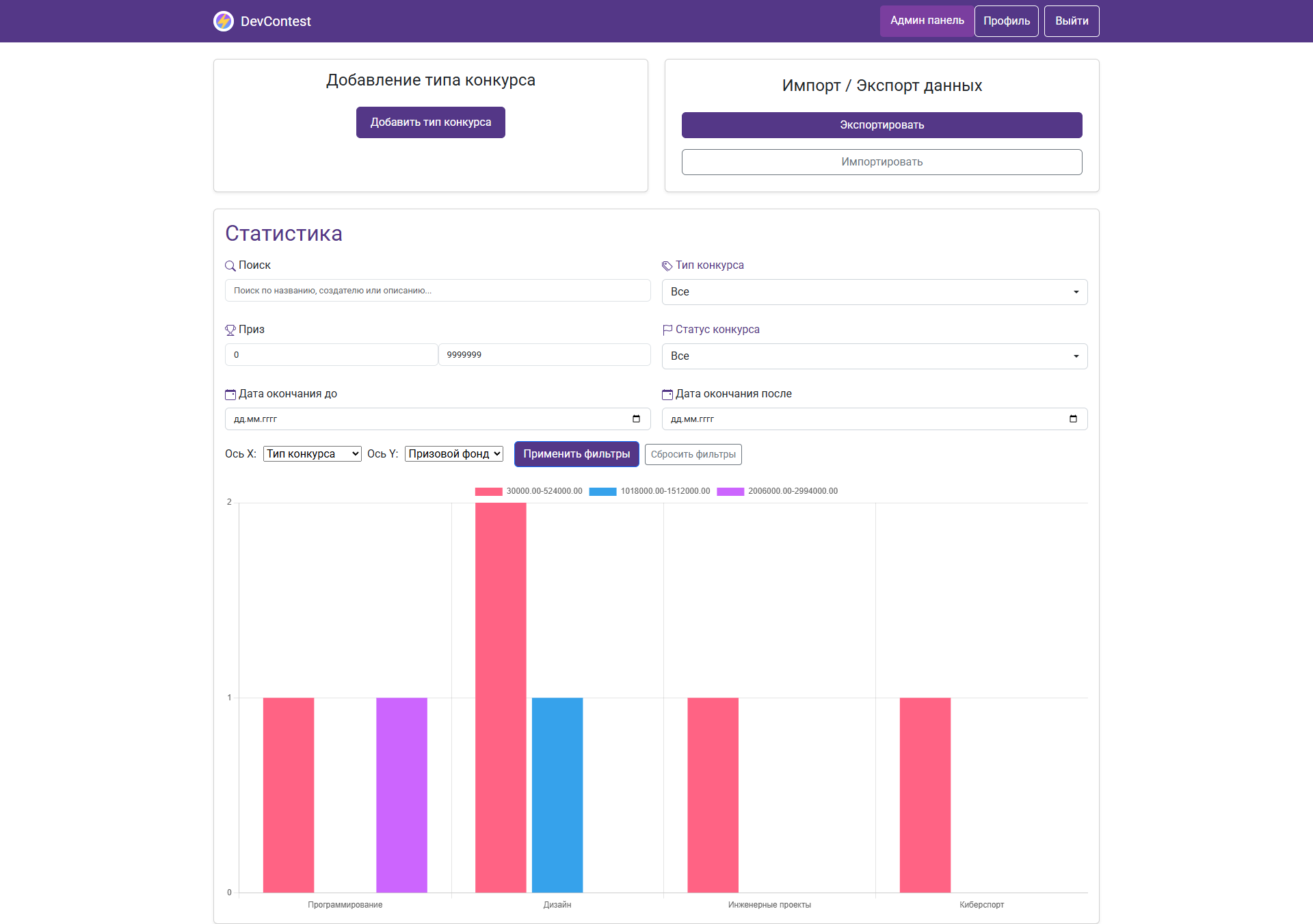


Рисунок 11 - Админ панель

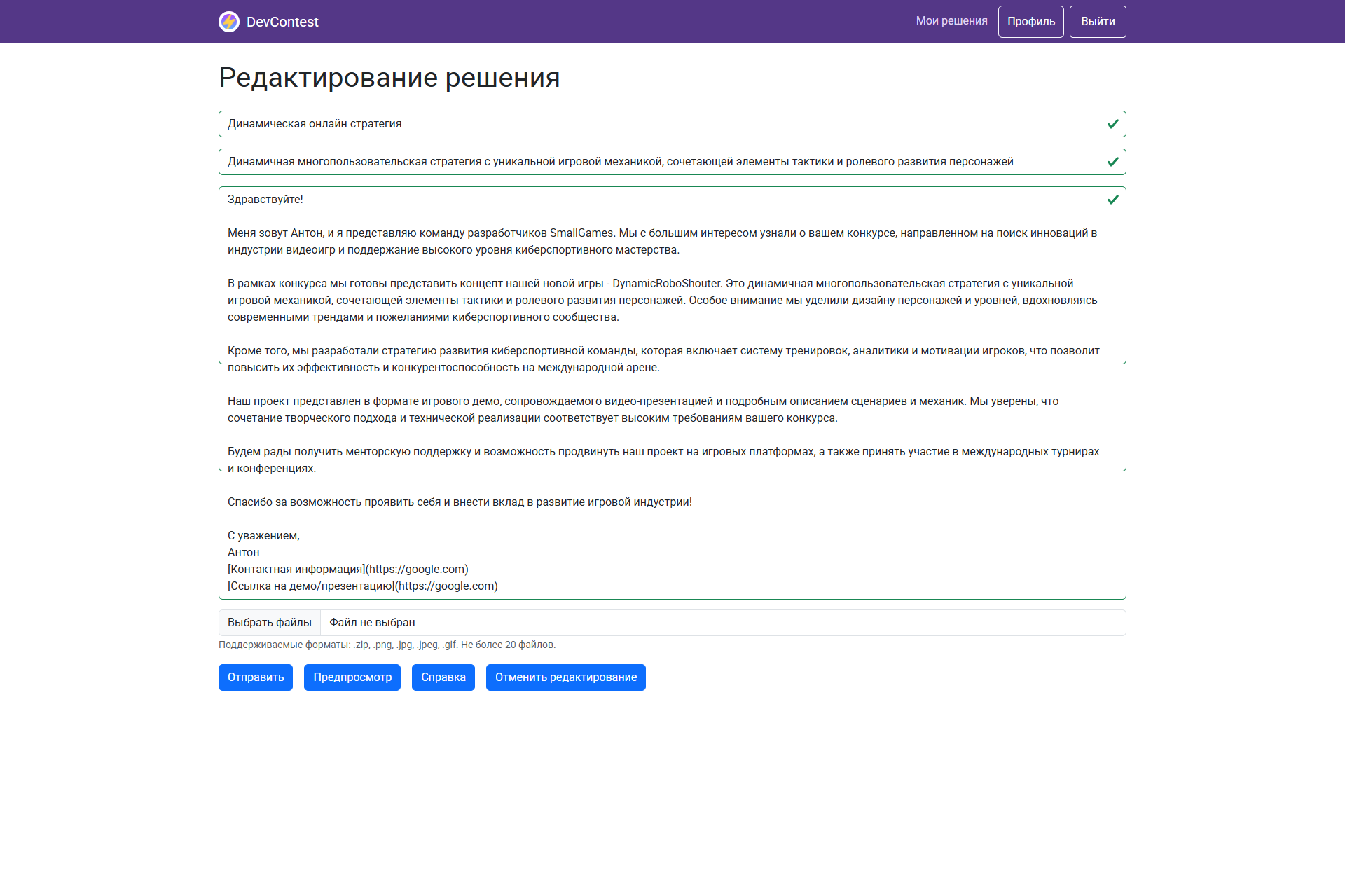


Рисунок 12 - Редактирование решения

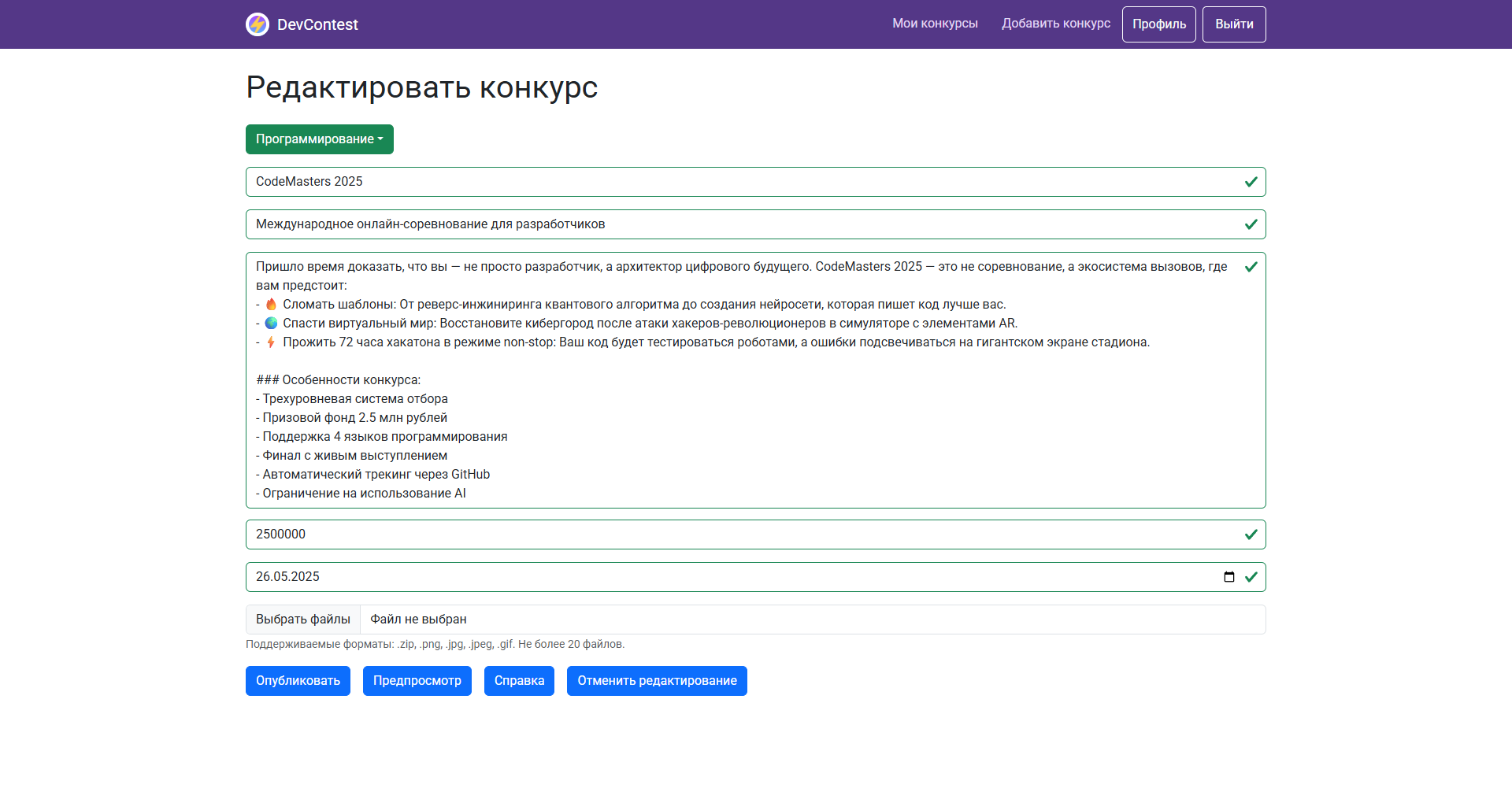


Рисунок 13 - Редактирование конкурса

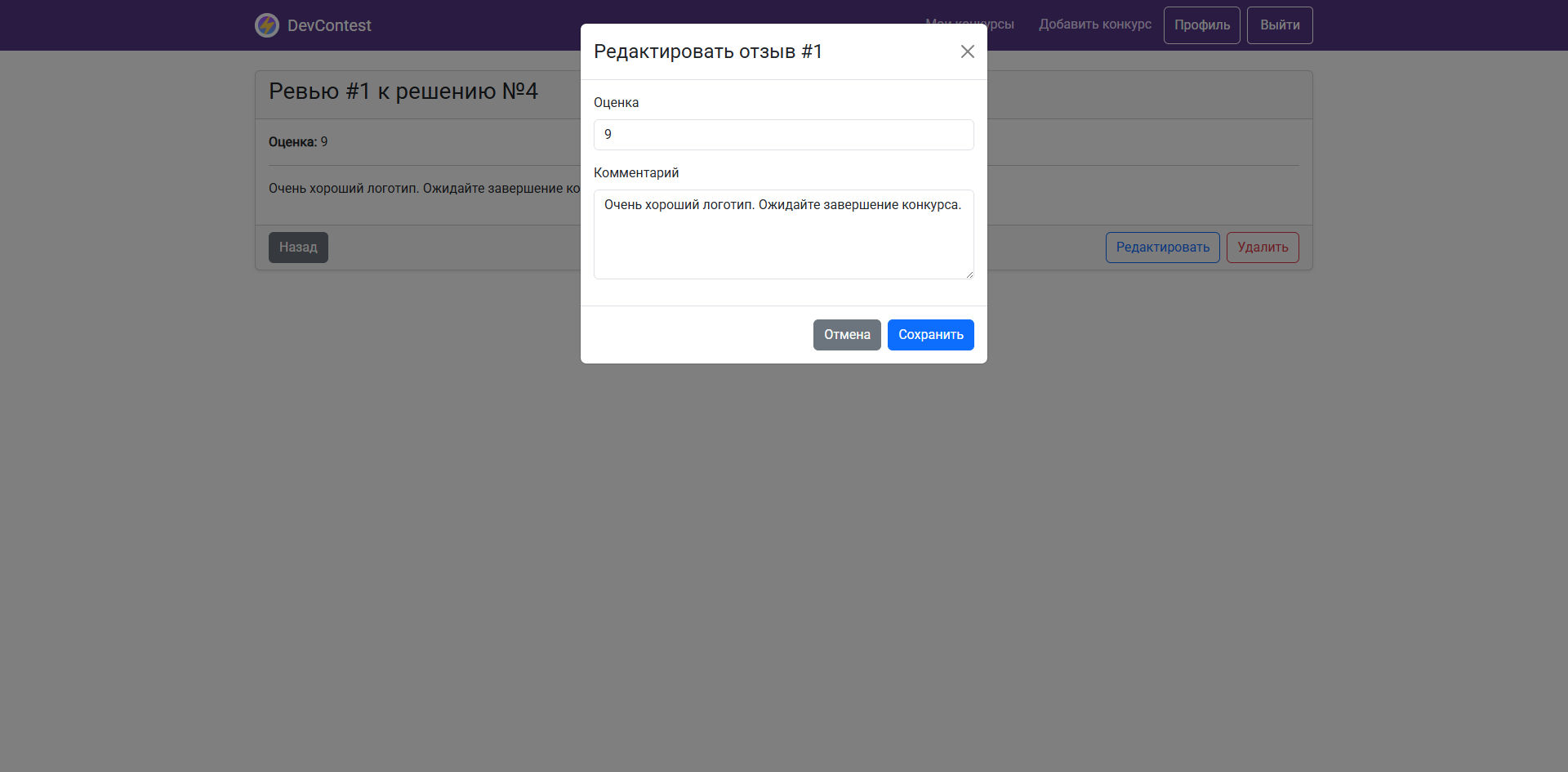


Рисунок 14 - Редактирование отзыва

**ЗАКЛЮЧЕНИЕ**

В рамках проекта была разработана и реализована веб-платформа DevContest — система для проведения конкурсов по выполнению креативных и технических задач. Приложение построено на клиент-серверной архитектуре с использованием современных технологий: React (фронтенд), Flask (бэкенд), MongoDB (база данных) и Docker (контейнеризация). Система обеспечивает удобную работу с конкурсами для трёх категорий пользователей: заказчиков, фрилансеров и администраторов.

Достигнутые результаты:

* Реализована удобная регистрация и авторизация пользователей с разными ролями.
* Спроектирована и внедрена нереляционная модель хранения данных с использованием MongoDB.
* Обеспечена полная функциональность CRUD-операций для конкурсов, решений и отзывов.
* Разработан интуитивно понятный пользовательский интерфейс с учётом UX-требований.
* Внедрён механизм импорта и экспорта данных в машиночитаемом формате.
* Проект успешно развернут в среде Docker и готов к эксплуатации.

Выявленные недостатки и пути улучшения:

* Отсутствует система уведомлений (email/внутренние оповещения) о статусе конкурсов и решений
* Не реализована система оценки активности пользователей или рейтинга, что может быть полезно для создания конкурентной среды.
* Интерфейс администратора можно расширить аналитикой по конкурсам и пользователям.
* Отсутствует многоязычная поддержка

Перспективы дальнейшего развития:

* Добавление возможности командного участия в конкурсах и реализации чатов для взаимодействия участников.
* Внедрение системы безопасных платежей и автоматизации выплаты призов победителям.
* Реализация адаптивной мобильной версии интерфейса.
* Интеграция с популярными социальными сетями и профессиональными платформами для упрощения входа и повышения доверия.
* Разработка панели модерации контента и автофильтрации нежелательной активности.

Таким образом, достигнута поставленная цель разработки базовой версии платформы DevContest. Полученное решение может быть использовано как MVP.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. React. Официальная документация. – URL: https://reactjs.org/ (дата обращения: 15.05.2025).
2. Flask. Официальная документация. – URL: https://flask.palletsprojects.com/ (дата обращения: 14.05.2025).
3. MongoDB. Официальная документация. – URL: https://www.mongodb.com/docs/ (дата обращения: 20.05.2025).
4. Docker. Документация по Docker и Docker Compose. – URL: https://docs.docker.com/ (дата обращения: 20.05.2025).
5. MobX. Документация по управлению состоянием в React-приложениях. – URL: https://mobx.js.org/ (дата обращения: 14.05.2025).
6. Axios. Официальная документация. – URL: https://axios-http.com/ (дата обращения: 20.05.2025).
7. Pydantic. Data validation and settings management using Python type annotations. – URL: https://docs.pydantic.dev/ (дата обращения: 20.05.2025).
8. Репозиторий проекта в GitHub URL:<https://github.com/moevm/nosql1h25-freelance>

**ПРИЛОЖЕНИЕ А**

**ДОКУМЕНТАЦИЯ ПО СБОРКЕ И РАЗВЕРТЫВАНИЮ**

Инструкция по сборке и запуску проекта

1. Клонируйте репозиторий проекта:

git clone https://github.com/moevm/nosql1h25-freelance

cd nosql1h25-freelance

2. Выполните сборку и запуск с помощью docker compose:

docker compose up --build

3. После запуска:

- клиентская часть будет доступна по адресу http://localhost:3000

- серверная часть — по адресу http://localhost:8000

4. Чтобы остановить и удалить контейнеры:

docker compose down

**ПРИЛОЖЕНИЕ Б**

**ИНСТРУКЦИЯ ДЛЯ ПОЛЬЗОВАТЕЛЯ**

Данная инструкция предназначена для пользователей веб-платформы DevContest, предназначенной для проведения конкурсов по выполнению креативных и технических задач.

1. Регистрация и авторизация

1.1. Перейдите на главную страницу сайта.

1.2. Нажмите на кнопку «Регистрация».

1.3. Заполните все поля формы (почта, логин, пароль).

1.4. Подтвердите регистрацию.

1.5. Для входа используйте кнопку «Вход» и укажите логин и пароль.

2. Работа заказчика

2.1. Для создания конкурса нажмите «Добавить конкурс».

2.2. Заполните поля: описание проекта, тип, срок окончания, приз.

2.3. Нажмите «Опубликовать конкурс».

2.4. Для управления конкурсами доступны функции редактирования, выбора победителя и удаления.

3. Работа фрилансера

3.1. Используйте фильтры на главной странице для выбора подходящих конкурсов.

3.2. Выберите интересующий конкурс, откройте его карточку.

3.3. Нажмите «Добавить решение», заполните описание, прикрепите файл и нажмите «Отправить».

3.4. Управление отправленными решениями осуществляется в разделе «Мои решения».

4. Панель администратора

4.1. Администратор может просматривать все конкурсы и решения.

4.2. Доступны функции удаления конкурсов и решений.

4.3. Импорт/экспорт данных осуществляется через панель администратора в формате ZIP/JSON.

5. Отображение данных

5.1. Конкурсы и решения отображаются в виде карточек с подробной информацией.

5.2. Пользователь видит только актуальные для своей роли элементы интерфейса.