0.1 Angular

Для выполнения работы потребовалось с фреймворком Angular по причине того, что веб-интерфейс пользователя был написан с использованием этой технологии.

Angular - это TypeScript-фреймворк с открытым исходным кодом, который предназначен для разработки одностраничных приложений. Цель фреймворка - расширение браузерных приложений на основе MVC-шаблона, а также упрощение разработки и тестирования.

Фреймворк работает с HTML, содержащим дополнительные пользовательские атрибуты, которые описываются директивами, и связывает ввод или вывод области страницы с моделью, представляющей собой обычные переменные TypeScript. Значения этих переменных задаются вручную или извлекаются из статических или динамических JSON-данных.

Базовой единицей в данном фреймворке является компонент, который содержит HTML шаблон, стили и логику в TypeScript файлах.

0.2 Markdown

Markdown — облегчённый язык разметки, созданный с целью написания наиболее читаемого и удобного для правки текста, но пригодного для преобразования в языки для продвинутых публикаций.

Markdown используется как язык разметки для статей базы знаний.

1 Расширение фунционала «Базы знаний»

1.1 Изучение предметной области

База знаний в данной системе представлена списком статей в формате Markdown, который упрощает форматирование текста. Изначально добавлять, редактировать статьи и их перевод мог только администратор, что наклады-

вало на базу ряд ограничений: трудность поддержки актуальности статей, их качественное и количественное содержание.

Поэтому было решено разрешить пользователям предлагать свои правки и переводы и, соответственно, добавить этот функционал в веб-интерфейс и сервер.

1.2 Добавление пользовательской формы для внесения правок

В качестве Markdown редактора был выбран SimpleMDE, так как он уже применялся в интерфейсе администратора для тех же целей. Редактор доступен только авторизованным пользователям, которые нажали на соответствующую кнопку. Если же пользователь не авторизован, то нужно показать форму входа, как на рисунке 1.1.

Также требуется реализовать возможность выбор языка локализации.

Примерный интерфейс пользователя представлен на рисунке 1.2.

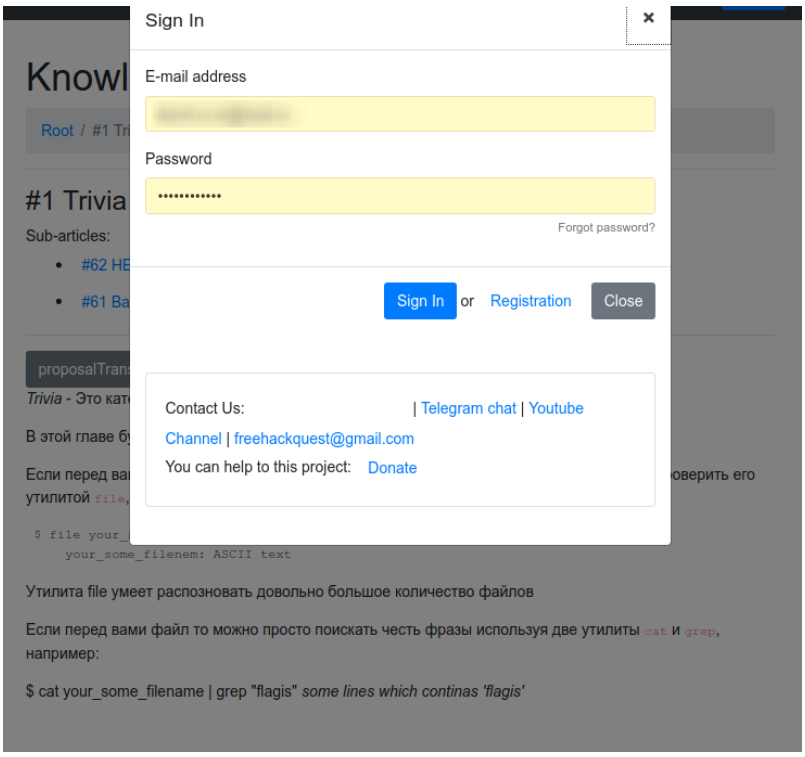

Рисунок 1.1 – Форма входа

По нажатию кнопки «Отправить» на сервер уходит запрос типа

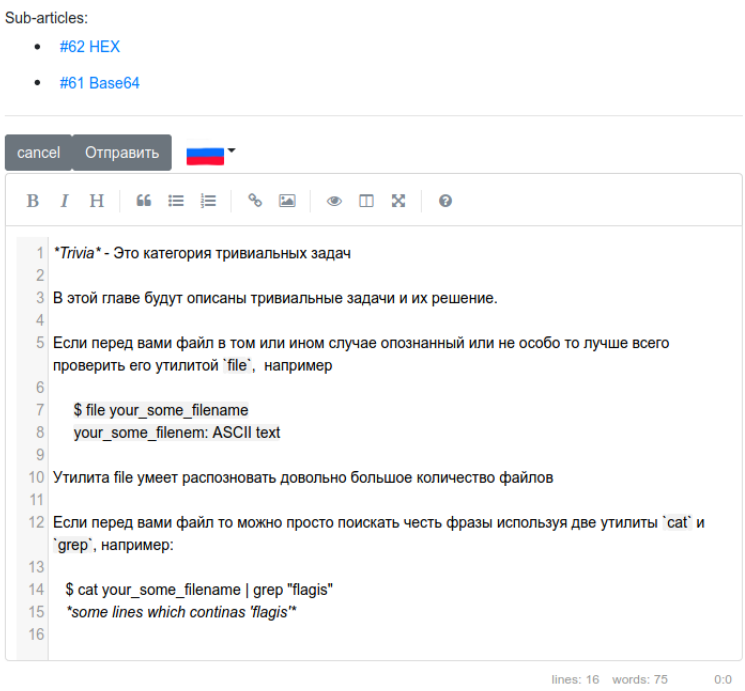

Рисунок 1.2 – Интерфейс редактора

classbook_proposal_add_record, в котором содержится предлагаемое содержимое, язык, идентификатор статьи.

Если добавление правки в базу данных прошло успешно, то на веб-интерфейсе отобразится сообщение, показанное на рисунке 1.3

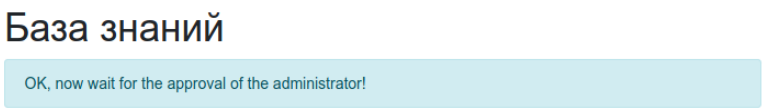

Рисунок 1.3 – Сообщение пользователю

1.3 Принятие и отклонение правок пользователей администратором

Что касается интерфейса администратора, то нужно было добавить отображение списка правок для данной статьи с отображением языка, возможностью редактировать, принять или отклонить правку. Причём кнопка Approve должна отображаться только для предложений.

Новый интерфейс администратора представлен на рисунке 1.4.

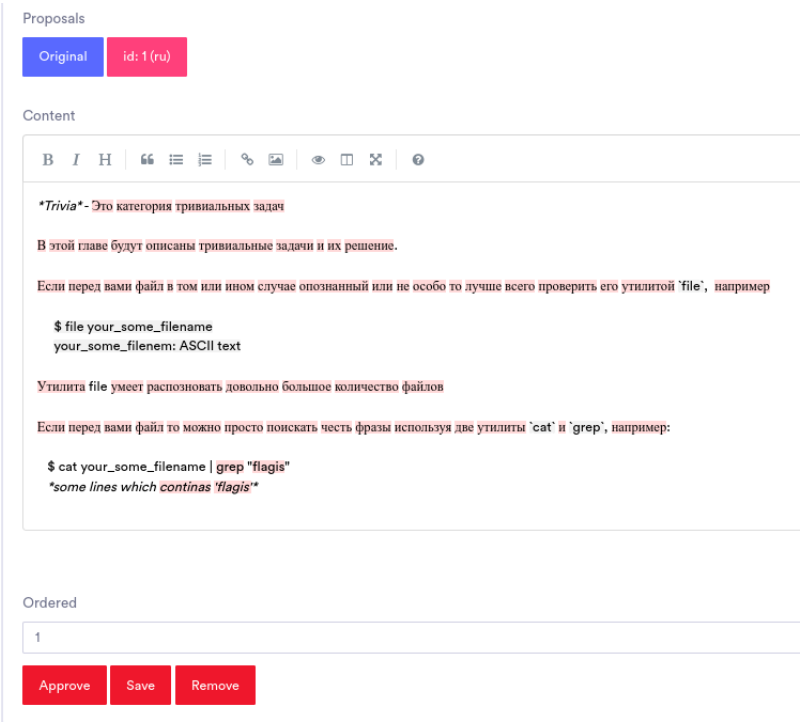

Рисунок 1.4 – Интерфейс администратора

Текущий выбор варианта статьи (оригинал или правка) для редактирования отображается другим цветом (розовым в данном случае) в списке «Propasals».

Данный список подгружает с сервера запросом classbook_proposal_list, который принимает идентификатор статьи.

Кнопка «Approve» отвечает за принятие правки, с помощью запроса classbook_proposal_approve.

Кнопка «Save» обновляет предложение или оригинальную статью, с помощью запроса classbook_proposal_update или classbook_update_record соответственно.

Кнопка «Remove» удаляет предложение или статью вообще, с помощью запроса classbook_proposal_remove или classbook_remove_record соответственно.

1.4 Серверная часть

Для функционала, описанного выше, были добавлены 2 новых типа запроса:
classbook_proposal_approve и classbook_proposal_update. Оба принимают идентификатор правки, а второе ещё и новое содержимое.

Структура базы данных, охватывающая 3 таблицы: classbook, classbook_proposal, classbook_localization, представлена на рисунке 1.5.

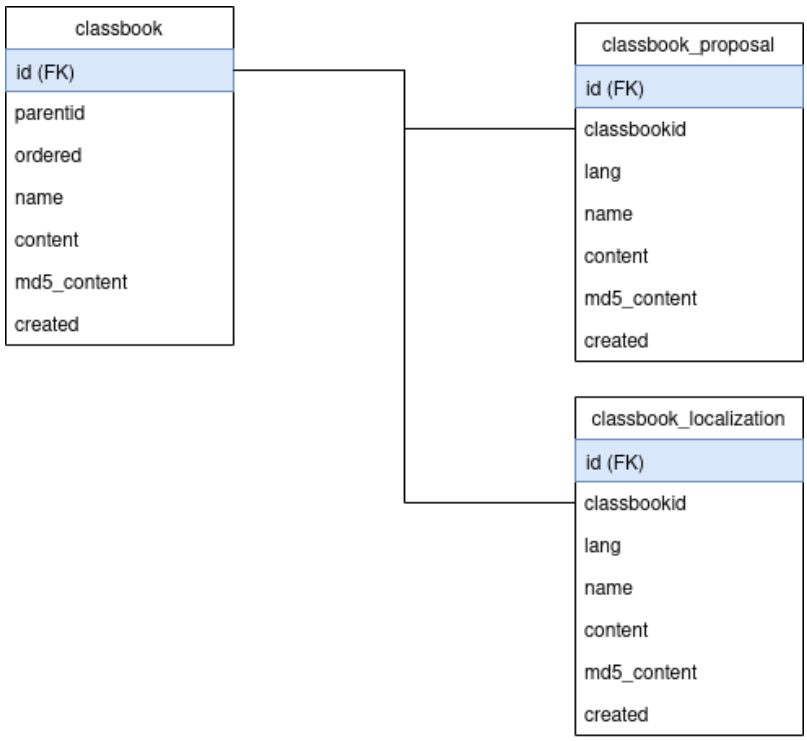

Рисунок 1.5 – Структура базы данных

В classbook хранятся метаинформация о статье (родительская статья, позиция, дата создания, md5 от контента) и само содержимое (имя, контент).

В classbook_localization хранятся данные локализации(язык, имя, контент) и отношения к оригинальной статье (идентификатор статьи).

В classbook_proposal хранятся правка (язык, имя, контент) и само содержимое (имя, контент).

В classbook_proposal_approve нужно получить идентификатор правки и найти саму статью, код на рисунке 1.6 реализует это.

```
nlohmann::json jsonRequest = pRequest->jsonRequest();

int nClassbookProposalID = -1;
if (jsonRequest.find("classbook_proposal_id") != jsonRequest.end()) {
    nClassbookProposalID = jsonRequest.at("classbook_proposal_id").get<int>();
}

if (nClassbookProposalID == -1) {
    pRequest->sendMessageError(cmd(), WSJCppError(404, "This proposal doesn't exist"));
    return;
}

QSqlDatabase db = *(pDatabase->database());
QSqlQuery query(db);

query.prepare("SELECT classbookid, content FROM classbook_proposal WHERE id = :classbook_proposal_id");
query.bindValue(":classbook_proposal_id", nClassbookProposalID);
if (!query.exec()) {
    pRequest->sendMessageError(cmd(), WSJCppError(500, query.lastError().text().toStdString()));
    return;
}
if (!query.next()) {
    pRequest->sendMessageError(cmd(), WSJCppError(404, "This proposal doesn't exist"));
    return;
}
```

Рисунок 1.6 – Получение идентификатора статьи

В зависимости от языка происходит выбор в какую таблицу производить запись, код на рисунке 1.7 производит запись в таблицу classbook, если язык правки - русский:

```
if (sLang = "ru") {

    query.prepare("UPDATE classbook SET content = :content, md5_content = :md5_content WHERE id = :classbookid");
    query.bindValue(":classbookid", nClassbookID);
    query.bindValue(":content", QString::fromStdString(sContent));
    query.bindValue(":md5_content", QString::fromStdString(sContentMd5_));
    if (!query.exec()) {
        pRequest->sendMessageError(cmd(), WSJCppError(500, query.lastError().text().toStdString()));
        return;
    }

    nlohmann::json jsonResponse;
    pRequest->sendMessageSuccess(cmd(), jsonResponse);

    return;
}
```

Рисунок 1.7 – Обновление статьи

Обновление правки в запросе classbook_proposal_update происходит в 2 этапа:

– Валидация входных данных - не отличается от соответствующего кода в предыдущем запросе.

– Запись в БД, которая представлена на рисунке 1.8

```
query.prepare("UPDATE classbook_proposal SET content = :content, md5_content = :md5_content WHERE id = :classbook_proposal_id");
query.bindValue(":classbook_proposal_id", nClassbookProposalID);
query.bindValue(":content", QString::fromStdString(sContent));
query.bindValue(":md5_content", QString::fromStdString(sContentMd5_));
if (!query.exec()) {
    pRequest->sendMessageError(cmd(), WSJCppError(500, query.lastError().text().toStdString()));
    return;
}

nlohmann::json jsonResponse;
pRequest->sendMessageSuccess(cmd(), jsonResponse);
```

Рисунок 1.8 – Обновление правки