МУНИЦИПАЛЬНОЕ БЮДЖЕТНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
СРЕДНЯЯ ОБЩЕОБРАЗОВАТЕЛЬНАЯ ШКОЛА №84
(МБОУ СОШ № 84)

Проектная работа

«Создание сайта Mathelper»

по информатике

Выполнила:
Орехова Варвара Егоровна,
10 «А» класс.

Руководитель:
Сидоркина Анастасия Сергеевна,
учитель информатики.

Воронеж, 2024

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

На данный момент я усиленно занимаюсь математикой и информатикой и хотела бы связать свою жизнь с IT индустрией. По мере того как я занималась олимпиадной математикой, я заметила, что сайтов с идеями и методами решения задач почти нет, поэтому я решила создать такой сайт, назвав его Mathelper (помощник по математике). Этот проект поможет людям, интересующимся математикой, систематизировать свои знания о предмете.

Актуальность проекта: в современном мире, где технологии играют ключевую роль в образовании и обучении, создание сайта с математическими фактами представляет собой актуальный и важный проект. Математика не только является фундаментом для понимания мира, но и обладает красотой и глубиной, которые могут быть представлены в увлекательном и доступном формате через веб-сайты. Проект по созданию сайта с математическими фактами может служить мостом между традиционным обучением и цифровым образованием, делая математику более доступной и интересной для широкой аудитории. Это особенно актуально в эпоху, когда люди все больше обращаются к онлайн-ресурсам для обучения и развлечения.

Цель проекта: создание сайта в помощь всем, изучающим математику.

Задачи:

1. научиться владеть React, JS, CSS, Docker, Yandex Cloud, Github;
2. научиться писать код для веб-сайтов;
3. выбрать дизайн сайта;
4. найти информационное наполнение сайта;
5. опубликовать сайт в сети Интернет.

Объект исследования: сайт по математике.

Предмет исследования: процесс разработки и создания математического сайта.

Методы исследования: изучение, анализ литературы, синтез, создание.

Ожидаемые результаты: сайт Mathelper с математическими идеями и методами решения олимпиадных задач.

Практическая значимость работы: Предоставить ценную информацию всем, изучающим математику, а также самой научиться создавать тематические веб-сайты.

Сроки реализации работы: 2023-2024 учебный год.

ГЛАВА 1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Веб-разработка – это всеобъемлющий процесс, который включает в себя множество этапов, которые обязательно нужно уметь делать будущему разработчику и программисту.

1.1 Vite - инструмент сборки

На данный момент для создания сайта удобно использовать инструмент сборки **Vite**, цель которого - обеспечить более быструю, бережливую и экономичную разработку современных веб-проектов. Vite самодостаточный и имеет практичные настройки по умолчанию, но он также может очень гибко настраиваться и расширять свой функционал.

Для создания первого проекта Vite требует установить **Node.js** - это платформа с открытым исходным кодом для работы с языком JavaScript. **JavaScript** - это язык программирования, который позволяет создавать динамически обновляемый контент, управлять мультимедиа и анимировать изображения. Это язык, который позволяет отображать периодически обновляемый контент, интерактивные карты, анимацию 2D/3D графики.

Vite поддерживает много шаблонов, на которых могут быть написаны сайты. В данном проекте будет использоваться **React** - это библиотека для языка программирования JavaScript с открытым исходным кодом для разработки пользовательских интерфейсов. Она помогает быстро и легко реализовать реактивность - явление, когда в ответ на изменение одного элемента меняется остальные элементы. Данная библиотека позволяет собирать сложный пользовательский интерфейс из маленьких изолированных кусочков кода, называемых «компонентами».

Чтобы создать каркас сайта с помощью инструмента сборки Vite, необходимо использовать **create vite** - это инструмент для быстрого старта проекта из основного шаблона для популярных фреймворков. Также понадобится пакетный менеджер для JavaScript, стандартный - **npm**, однако сейчас его заменяют более современным пакетным менеджером **Yarn**. Для

того чтобы установить пакетный менеджер Yarn, в терминале надо написать: yarn install. Затем создаем сайт, написав в терминале: yarn create vite myApp --template react. После ввода команды появится папка “myApp”. В ней хранятся нужные для сайта файлы. Далее, для того чтобы запустить сайт локально, т.е. на нашем компьютере, пишем в терминале команду yarn run dev. После этого терминал заморозится, и появится ссылка с локальным хостом и портом, на котором запущен сайт. Перейдя по ней, мы попадем на сайт. На данный момент это просто базовая страница Vite + React. Структура сайта готова. **(Приложение 1)**

1.2 Работа с файлами и кодом

Важно понимать иерархию файлов: в проекте Vite index.html является центральным файлом. Во время разработки Vite является сервером, а index.html - точкой входа в приложение. Vite рассматривает index.html как исходный код. Он разрешает скрипт, который ссылается на исходный код JavaScript, т.е. index.html ссылается на main.jsx. А main.jsx ссылается на компонент App, т.е. компонент, в котором написано, как будет выглядеть сайт. Далее можно создавать различные компоненты и экспортировать их в App или в другие компоненты. **(Приложение 2)**

Теперь вместо кода, существующего по умолчанию, следует написать собственный код. Для верстки сайта используется **HTML** - язык разметки, который используется для визуального и смыслового структурирования нашего контента, например, для определения параграфов, заголовков, таблиц данных, или вставки изображений и видео на страницу, **CSS** - язык стилей, с помощью которого определяется стиль нашего HTML контента, например, решается цвет фона и шрифта, или расположение текста: слева, справа или посередине, а также **JavaScript** - язык программирования, который позволяет создавать динамически обновляемую информацию. Вместе с JavaScript работает библиотека React, поэтому расширение файлов - **JSX**.

1.3 React. Компоненты. React Router

Так как пишем код на **React**, важно понимать его преимущество: данная библиотека позволяет собирать сложный пользовательский интерфейс из маленьких изолированных элементов кода, называемых «компонентами». **Компоненты** помогают сделать код многоразовым. Они позволяют разбить интерфейс на независимые части. Их можно соединять вместе и использовать несколько раз. Компоненты обязательно пишутся с большой буквы, проще всего объявить их через функцию. В конце файла компонент нужно обязательно экспортировать, а там, где мы собираемся его использовать, нужно импортировать. Компонент-функция получает данные в одном объекте («пропсы») в качестве параметра и возвращает React-элемент. Поэтому весь код не будет написан в одном файле, а будет разделен на множество различных компонентов, чтобы удобнее было с ними работать.

Теперь можно приступать к написанию кода для проекта. React позволяет пользоваться **React Router** - это библиотека, которая играет роль путеводителя в мире React приложений. Чтобы начать работу с React Router, необходимо установить пакет react-router-dom в проект, в терминале написав: yarn add react-router-dom. После установки пакета можно начать определять маршруты на сайте. Основные элементы, с которыми предстоит работать:

<Router>: оболочка вокруг приложения, которая использует HTML5 history API для синхронизации пользовательского интерфейса с ссылкой.

<Route>: компонент, который отображает пользовательский интерфейс, если его путь совпадает с текущей ссылкой. **(Приложение 3)**

1.4 Дизайн сайта. База данных

Следующий этап - создание дизайна сайта. Для этого пишется код с помощью языка CSS. Необходимо создать файл с расширением .css и прописать там все стили, пользуясь правилами оформления CSS кода.

Затем следует создать базу данных. В данном проекте она будет находиться в JSON файле, т.к. он позволяет хранить массив данных и передавать его. Создаем файл с расширением .json, в нем создаем массив данных, где каждый элемент будет иметь имя, тему, описание и id. Потом добавляем туда информацию. **(Приложение 4)**

1.5 Хостинг сайта. GitHub. Docker

Когда код написан, итогом является сайт, который работает только на локальном компьютере. Следующий этап - размещение в сети Интернет. Для этого понадобиться **хостинг** - сервис для размещения сайтов, там можно приобрести удаленный сервер, постоянно подключенный к интернету и работающий круглосуточно, на котором находятся все файлы сайта. Нам также понадобится **домен или ip-адрес**. Все интернет-ресурсы пользуются доменами, но в данном проекте будет использован ip-адрес. Часто сайты предоставляют бесплатный хостинг на определенное время, например Yandex Cloud, там же дадут технический ip-адрес, для того чтобы открыть сайт в браузере. Заходим на сайт Yandex Cloud и, пользуясь инструкциями, создаем виртуальную машину, выбирая необходимые настройки. На данной машине размещаем данный сайт.

Для того чтобы хранить код в интернете, следует воспользоваться **GitHub** - сервисом для совместной разработки и хостинга проектов. С помощью GitHub над кодом проекта может работать неограниченное количество программистов из любых точек мира. GitHub - это веб-сервис для хостинга git-репозиториев. **Git** - это распределенная система управления версиями, позволяющая просматривать и контролировать любые изменения кода любым разработчиком и возвращаться к первоначальному состоянию. В целом, это социальная сеть для разработчиков, в которой можно найти проекты с открытым кодом от других разработчиков, практиковаться в написании кода и хранить свое портфолио. Пользуясь инструкциями сайта,

создаем аккаунт и репозиторий на GitHub, чтобы хранить там данный проект. **(Приложение 5)**

Существует проблема: сайт может не работать при условии, что версии программ разработчика и пользователя не совпадают. Тогда на помощь приходят **Контейнеры**, они позволяют упаковать в единый образ приложение и все его зависимости: библиотеки, системные утилиты и файлы настройки. Это упрощает перенос приложения на другую инфраструктуру. Чтобы использовать контейнеры, необходимо воспользоваться платформой **Docker** - это программная платформа для разработки, доставки и запуска контейнерных приложений. Она позволяет создавать контейнеры, автоматизировать их запуск и развертывание, управляет жизненным циклом.

Таким образом, есть возможность создать контейнеры для базы данных и фронтенда проекта, тогда они будут запускаться автоматически. Важно понимать, что контейнер - это набор процессов, изолированных от основной операционной системы.

ГЛАВА 2. ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 Этапы создания сайта

Создание сайта предполагает наличие нескольких этапов. Часто порядок их выполнения зависит от целей проекта. В данной работе было важно научиться создавать сайт, поэтому в первую очередь был написан код. Однако профессионалы начинают с идеи - схематичного наброска, где указывают основные блоки, после этого следует перевод графического изображения на язык, понятный браузеру, т.е. верстка. Создавая свой сайт, я придерживалась такого плана:

1. выбрать систему управления сайтом или конструктор;
2. продумать структуру и навигацию сайта;
3. дизайн сайта;
4. наполнение сайта контентом;
5. хостинг сайта.

Мною было принято решение написать сайт на **React** с помощью инструмента сборки **Vite**, использовать файлы с расширением **JSX**. Для дизайна использовать **CSS**. Контент сайта, математические факты, идеи и методы решения задач, возьму из книги **В. В. Ткачука "Математика абитуриенту"**. Все процессы запустить на **Docker**, а хостить сайт на **Yandex Cloud**. Хранить файлы с кодом на **GitHub**.

2.2 Выбор системы управления сайтом

Вначале надо было выбрать систему управления сайтом. Потом создать с помощью нее шаблон, который необходимо взять за основу сайта. Мною были выполнены следующие этапы работы:

1. Установка пакетнего менеджер **Yarn**. Команда в терминале: `yarn install`.
2. Создание каркаса сайта с помощью инструмента сборки **Vite** на **React**. Команда в терминале: `yarn create vite myApp --template react`.

3. Запуск сайта. Команда в терминале: yarn run dev.

   После появления ссылки с локальным хостом и портом переходим по ней, чтобы попасть на сайт и удаляем код, существующий по умолчанию. **(Приложение 1)**

2.3 Структура и навигация сайта. Создание фронтенда

Далее надо было написать код для фронтенда, пользуясь **React**. Мною была разработана структура сайта и созданы следующие компоненты:

- **App** - главный компонент. В данном компоненте прописаны все возможные пути пользователя и их исходы, т.е. как будет выглядеть сайт в определенный момент.
- **Navbar** - компонент навигационной панели. Здесь прописаны строки, которые будет отображаться сверху. На навигационной панели расположены три кнопки: “домой”, “создать новый факт” и “Mathelper”, которое также является ссылкой домой.
- **Home** - компонент “домой”. Тут прописан код, который будет отображаться на главной странице “домой”.
- **NotFound** - компонент страницы, которая будет появляться, если пользователь зайдет на несуществующую страницу сайта.
- **Create** - компонент, позволяющий создавать факты. Тут прописан код, с помощью которого пользователь создает факты прямо на сайте: он принимает данные и передает их в Json файл, где находится база данных.
- **FactDetails** - компонент страницы, которая будет появляться, если пользователь будет заходить на определенный факт.
- **FactList** - компонент, который выводит факты из базы данных. Данный компонент используется в компоненте **Home**, чтобы на главной странице появлялись факты.
- **useFetch** - это хук (в переводе с английского «hook» — ловушка), который позволяет вытягивать данные из нашей базы данных.

Далее мною был написан код для главного компонента App. Для этого были выполнены следующие действия:

1. Импорт библиотеки React, BrowserRouter и нужных компонентов.

2. Объявление компонента App в качестве функции (в данном случае - обычной). В данном компоненте прописываем все возможные пути: если человек будет переходить по пути

- “/”: на страницу будет передаваться компонент Home, и будет открываться главная страница “Домой”.
- “/create”: на страницу будет передаваться компонент Create, и будет открываться страница “Создать новый факт”..
- “/facts/:id”: на страницу будет передаваться компонент FactDetails, и будет высвечиваться страница с определенным фактом.
- “/*”: на страницу будет передаваться компонент NotFound, и будет выводиться страница, на которой ничего нет и на которой написано предложение вернуться на главную страницу.

3. Экспорт компонента App, чтобы позже суметь воспользоваться им.

Таким образом, мною была написана логика проекта на React. **(Приложение 2, 3)**

2.4 Дизайн сайта

Затем мною были прописаны стили сайта. Для этого мною был использован CSS. Сперва надо было создать файл index.css и в нем прописать стили для каждого компонента отдельно, пользуясь правилами CSS языка. Позже этот файл был импортирован в main.jsx. **(Приложение 4)**

2.5 Создание базы данных. Наполнение сайта контентом

Для создания базы данных был использован Json файл, т.к. он позволяет хранить массив данных. Мною была создана папка data, а в ней -

файл db.json, база данных. Наполнение сайта было взято из книги В. В. Ткачука “Математика Абитуриенту”. Каждый прописанный факт имеет имя, описание, тему, id.

2.6 Хостинг сайта

Затем я создала репозиторий для своего проекта на GitHub, сохранив там написанный код. **(Приложение 5)**. В терминал я написала следующее:

```
git add .

git commit -m “first commit”

git push -u origin main
```

Далее я воспользовалась Yandex Cloud, чтобы приобрести свою виртуальную машину, на которой будет размещен данный сайт на выданном мне ip-адресе, 84.252.136.128. Затем мною был сгенерирован ssh-ключ (на данный момент является самой безопасной альтернативой паролю) с помощью команды ssh-keygen. И я смогла зайти на свою виртуальную машину, в терминале написав: kitten ssh user@84.252.136.128. Затем склонировала свой репозиторий с кодом на свою виртуальную машину, написав в терминале: git clone <ssh-ссылка на наш репозиторий>. Теперь данный проект размещен на выданном мне ip-адресе. **(Приложение 6)**

Далее предстояла работа с Docker контейнерами. Мне надо было создать два контейнера: первый - для фронтенда (для всего логического кода), второй - для базы данных. Для начала мною было создано два Docker файла, которые нужны для записи образа. В них описывается, что должно находиться в образе, какие команды, зависимости и процессы он будет содержать. Я написала их, используя документацию Docker, а затем для сборки образа я использовала команду: docker build <докерфайл> и для запуска образов: docker run --name <образ> -p <ip-адрес> .

2.7 Итог сайта

Таким образом, мне удалось создать сайт Mathepler с математическими идеями и методами решения олимпиадных задач.

На главной странице сайта расположена навигационная панель и все математические факты. **(Приложение 7)**

На навигационной панели есть кнопка “создать факт”, нажав на нее, попадаешь на отдельную страницу для создания факта. **(Приложение 8)**

ЗАКЛЮЧЕНИЕ

В процессе выполнения проекта мною была проделана значительная работа: отбор литературы, поиск нужной информации, анализ данных и непосредственно создание сайта и наполнение его нужным контентом, а также размещение его в сети Интернет. Этот проект научил меня создавать сайты и помог людям, интересующимся олимпиадной математикой, получать новые знания на безвозмездной основе.

Ссылка на мой сайт: http://84.252.136.128:5173/

В процессе выполнения проекта, я реализовала поставленные передо мной задачи:

1. научилась владеть React, JS, CSS, Docker, Yandex Cloud, Github;
2. научилась писать код для веб-сайтов;
3. выбрала дизайн сайта;
4. нашла информационное наполнение сайта;
5. опубликовала сайт в сети Интернет.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ:

1. В. В. Ткачук “Математика абитуриенту” М.:МЦНМО 2018 г. 944 стр.
2. Docker. Документация URL: [https://docs.docker.com/]
3. DCMedia: статья “Что нужно уметь, чтобы создать сайт” URL:[https://dcmedia.ru/blog/chto-nuzhno-umet-chtoby-sozdat-sayt]
4. GitHub Docs. Документация URL: [https://docs.github.com/en/get-started]
5. Net Ninja “Курс для написания сайта на React”. Документация URL: [https://github.com/iamshaunjp/Complete-React-Tutorial]
6. React. Документация URL: [https://react.dev/]
7. Skillfactory Media. База знаний [https://blog.skillfactory.ru/knowledge-base/]
8. Skypro Media. Программирование [https://sky.pro/media/programmirovanie/]
9. Vite. Официальный сайт. Руководство [https://vite-docs-ru.vercel.app/guide/]
10. W3Schools. Туториалы для всех языков программирования URL:[https://www.w3schools.com/]

Приложение 1

```
➜ ~ yarn create vite myApp --template react
yarn create v1.22.22
[1/4] Resolving packages...
⠂ (node:64296) [DEP0040] DeprecationWarning: The `punycode` module is deprecated. Please use
a userland alternative instead.
(Use `node --trace-deprecation ...` to show where the warning was created)
[2/4] Fetching packages...
[3/4] Linking dependencies...
[4/4] Building fresh packages...

success Installed "create-vite@5.2.3" with binaries:
      - create-vite
      - cva
✔ Package name: … myapp

Scaffolding project in /home/brb/myApp...

Done. Now run:

  cd myApp
  yarn
  yarn dev

Done in 1.96s.
➜ ~ cd myApp/
➜ myApp yarn install
yarn install v1.22.22
info No lockfile found.
[1/4] Resolving packages...
⠂ (node:64387) [DEP0040] DeprecationWarning: The `punycode` module is deprecated. Please use
a userland alternative instead.
(Use `node --trace-deprecation ...` to show where the warning was created)
[2/4] Fetching packages...
[3/4] Linking dependencies...
[4/4] Building fresh packages...
success Saved lockfile.
Done in 19.65s.
➜ myApp yarn run dev
yarn run v1.22.22
$ vite

  VITE v5.2.10  ready in 456 ms

  ➜  Local:   http://localhost:5173/
  ➜  Network: use --host to expose
  ➜  press h + enter to show help
```

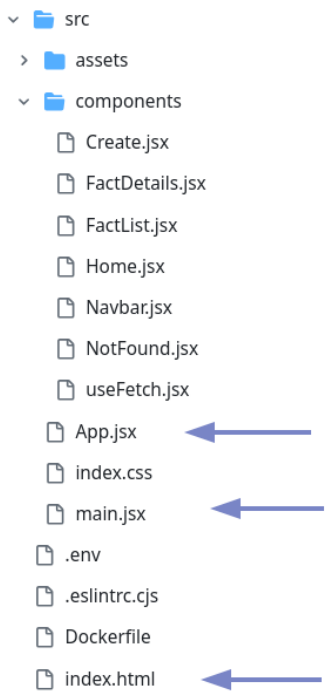
Vite + React

count is 0

Edit src/App.jsx and save to test HMR

Click on the Vite and React logos to learn more

Приложение 2



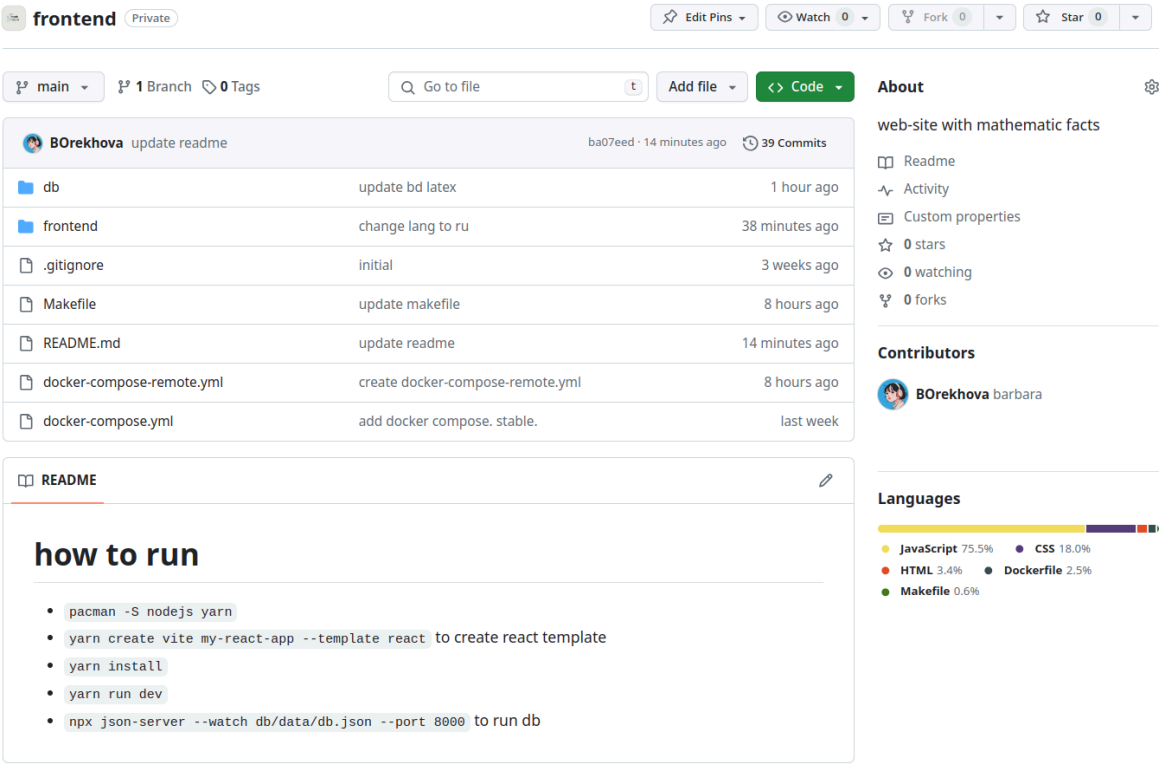
Приложение 3

```jsx
import React from 'react';

import Home from './components/Home';
import Navbar from './components/Navbar';
import Create from './components/Create';
import NotFound from './components/NotFound';
import FactDetails from './components/FactDetails';

import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';

function App() {
  return (
    <Router>
    <div className="App">
      <Navbar />
      <div className="content">
        <Switch>
          <Route exact path="/" component={Home} />
          <Route path="/create" component={Create} />
          <Route path="/facts/:id" component={FactDetails} />
          <Route path="/*" component={NotFound} />
        </Switch>
      </div>
    </div>
    </Router>
  )
}

export default App;
```

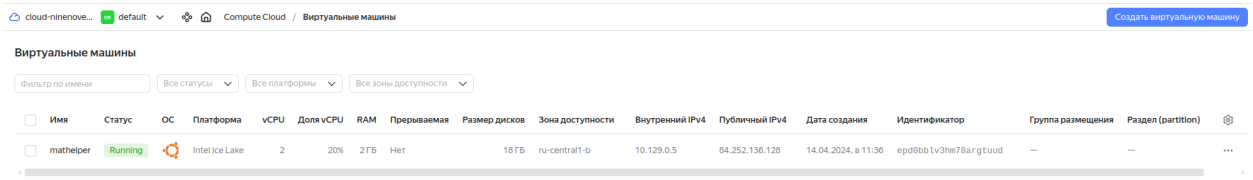
Приложение 4

```
{
  "facts": [
    {
      "title": "Уравнения и неравенства с модулями",
      "theme": "algebra",
      "body": "Для того, чтобы решить любую задачу с модулями, необходимо знать определение модуля числа a. Его можно дать,
      "id": "1"
    },
    {
      "title": "Связь геометрических и алгебраических интерпретаций",
      "theme": "algebra",
      "body": " Практически всегда при решении геометрических задач необходимо обращаться к использованию алгебры. Для этого
      "id": "3"
    },
    {
      "title": "Переход к новым переменным",
      "theme": "algebra",
      "body": "Соотношения в задаче обязательно записываются при использовании формул. От правильного использования формул з
      "id": "4"
    },
```

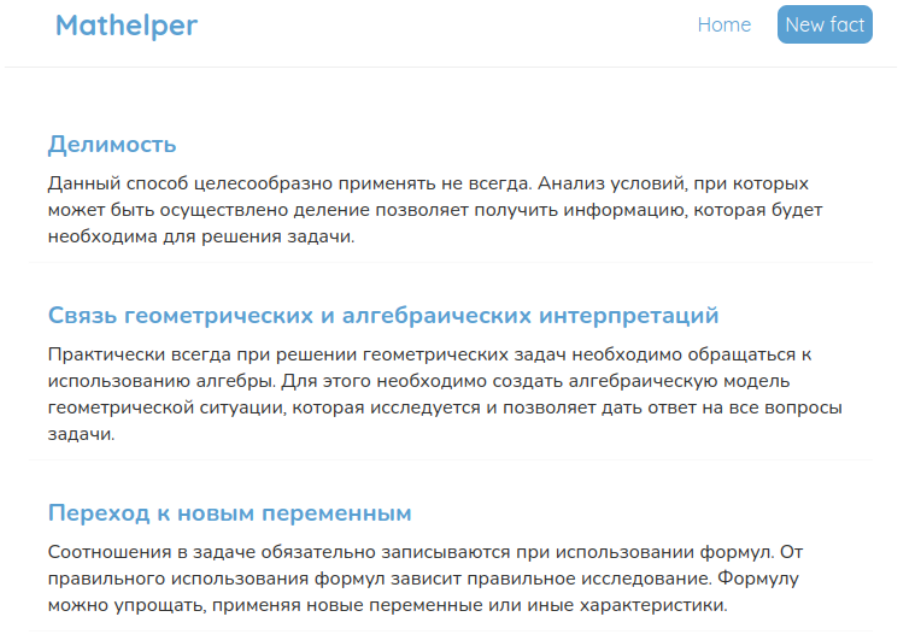

Приложение 5



Приложение 6



Приложение 7



Приложение 8