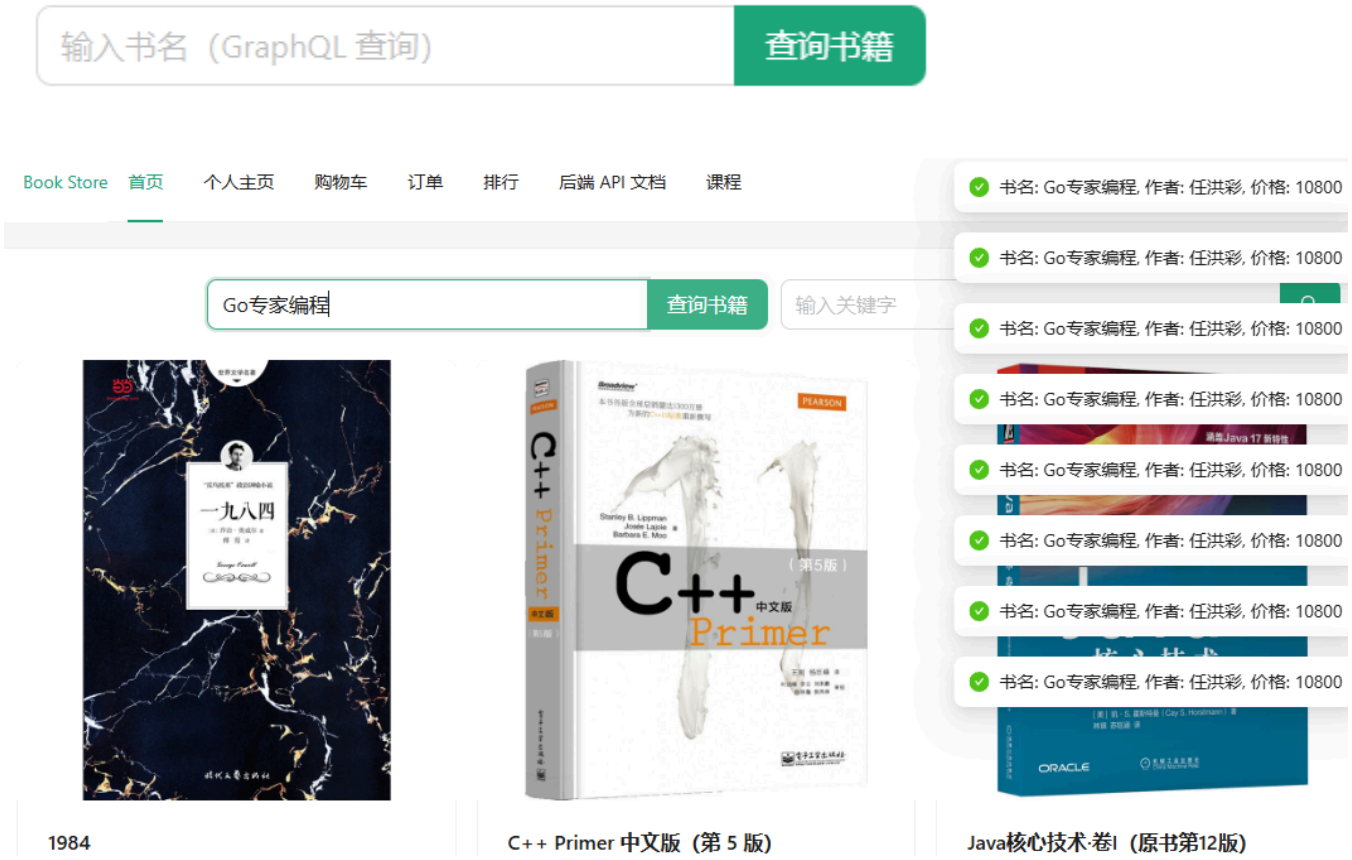
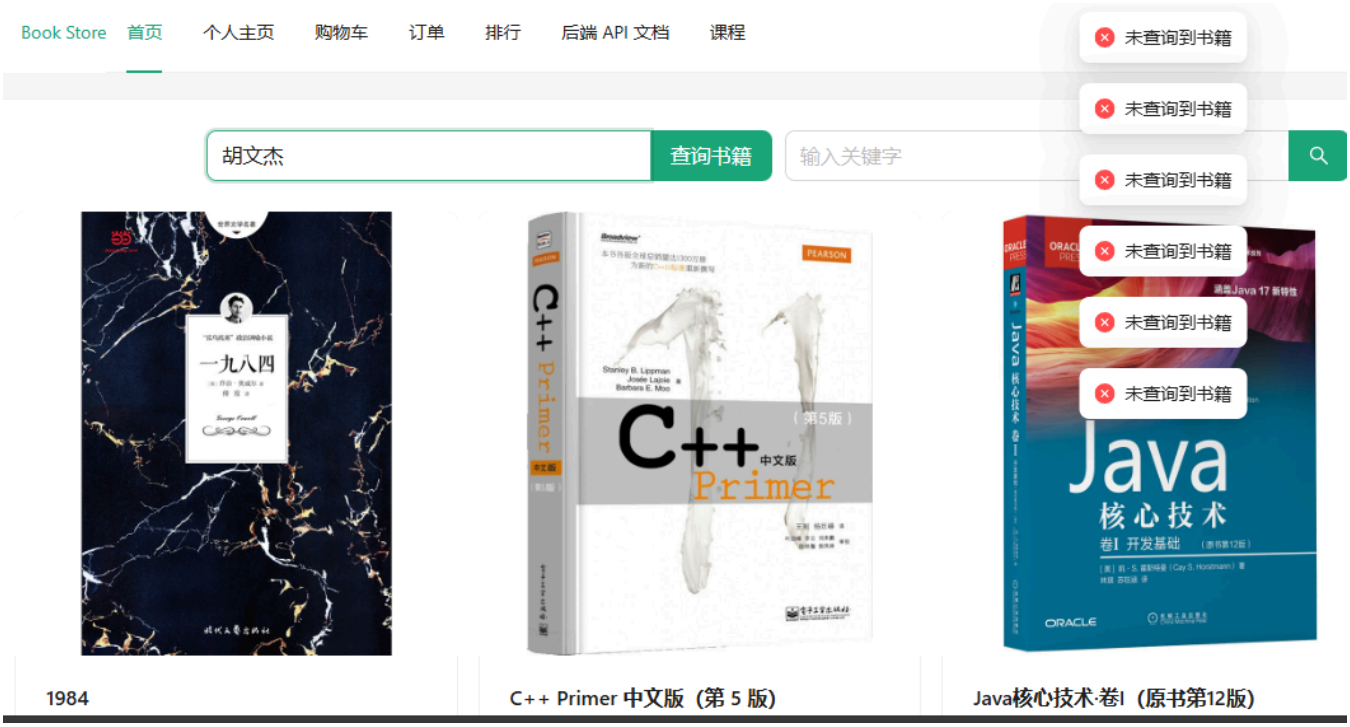
我做的比较简便，主要是前端输入书名，后端返回书名作者以及价钱。

前端

示例





前端代码修改

增加两个依赖

```
"@apollo/client": "^3.12.4",
"@testing-library/jest-dom": "^6.5.0",
"@testing-library/react": "^16.0.1",
"@testing-library/user-event": "^14.5.2",
"antd": "^5.21.1",
"antd-img-crop": "^4.23.0",
"graphql": "^16.10.0",
```

App 要用这玩意包起来

```tsx
import { ConfigProvider, theme } from "antd";
import AppRouter from "./components/router";
import { ApolloClient, InMemoryCache, ApolloProvider } from "@apollo/client";

const client = new ApolloClient({
  uri: "http://localhost:8070/graphql",
  cache: new InMemoryCache(),
});

function App() {
  const themeToken = {
    colorPrimary: "#1DA57A",
    colorInfo: "#1DA57A",
  };

  return (
    <ApolloProvider client={client}>
      <ConfigProvider
        theme={{
          algorithm: theme.defaultAlgorithm,
          token: themeToken,
        }}
      >
        <AppRouter />
      </ConfigProvider>
    </ApolloProvider>
  );
}

export default App;
```

前端发送 GraphQL 请求的

```
const handleGraphQLSearch = async (title) => {
  try {
    const response = await axios.post(
      `${process.env.REACT_APP_BASE_URL}/graphql`,
      {
        query: `
          query {
            getBookByTitle(title: "${title}") {
              title
              author
              price
            }
          }
        `,
      }
    );
    const book = response.data?.data?.getBookByTitle;
    if (book) {
      message.success(
        `书名: ${book.title}, 作者: ${book.author}, 价格: ${book.price}`
      );
    } else {
      message.error("未查询到书籍");
    }
  } catch (error) {
    message.error("查询失败");
  }
};
```

前端搜索框的样式

```
            <Col span={6}>
              <Search
                placeholder="输入书名 (GraphQL 查询) "
                onSearch={handleGraphQLSearch}
                enterButton="查询书籍"
                size="large"
              />
            </Col>
            <Col span={6}>
```

后端

引入依赖

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-graphql</artifactId>
</dependency>
```

controller 里面注意@Argument，可能会遇到跨域问题。

```java
package com.reins.bookstore.controller;

import com.reins.bookstore.dto.GraphQLBook;
import com.reins.bookstore.entity.Book;
import com.reins.bookstore.repository.BookRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.graphql.data.method.annotation.Argument;
import org.springframework.graphql.data.method.annotation.QueryMapping;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.CrossOrigin;

@Controller
@CrossOrigin(origins = "http://localhost:3000")
public class BookGraphQLController {
  @Autowired BookRepository bookRepository;

  @QueryMapping
  public GraphQLBook getBookByTitle(@Argument String title) {
    System.out.println("getGraphQLBookByTitle: " + title);

    Book book = bookRepository.findBookByTitle(title);
    if (book == null) {
      return null;
    }
    GraphQLBook graphQLBook = new GraphQLBook();
    graphQLBook.setTitle(book.getTitle());
    graphQLBook.setAuthor(book.getAuthor());
    graphQLBook.setPrice(book.getPrice());
    return graphQLBook;
  }
}
```

同步增加一个这个类

```java
package com.reins.bookstore.dto;

import lombok.Data;

@Data
public class GraphQLBook {
  private String title;
  private String author;
  private Integer price;
}
```

下面这个文件是在 resources 里面新建一个 graphql 文件夹，里面放一个叫 `schema.graphqls` 文件

```
type Query{
    getBookByTitle(title: String!): GraphQLBook
}

type GraphQLBook{
    title: String
    author: String
    price : Int
}
```