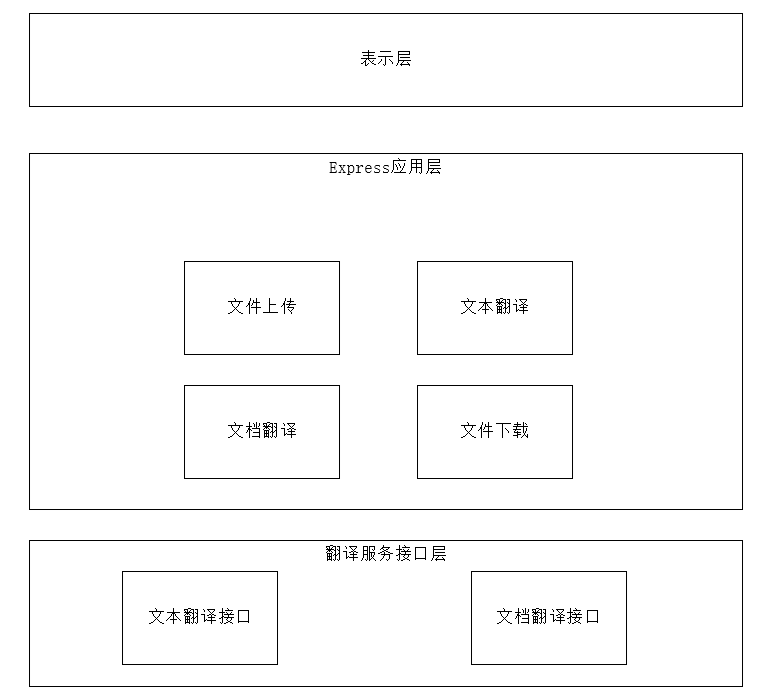
## 系统设计

### 架构设计

FileTrans采用分层架构的架构模式，架构示意图如下：



#### 表示层（Presentation Layer）

表示层作为用户与系统互动的桥梁，涵盖了用户界面、数据输入与输出等功能。在这一层，主要职责是将用户的请求有效地传递至系统的下一层级，并确保处理结果能够准确无误地反馈给用户。在我们的翻译系统中，表示层具体包括用户输入文本的界面、用户输入上传文本的界面、展示翻译结果的界面、以输出翻译内容的页面等。

#### Express应用层（Express Application Layer）

Express应用层是系统的核心层，它实现了翻译的核心算法和业务逻辑，包括文本处理、文本上传，文本下载等。这一层主要负责接收并处理表示层传递的请求，然后调用其他层的服务，最后将处理结果返回给表示层。在我们的WEB系统中，应用层负责：

1. 处理客户端：

* 文本输入
* 文档上传

2. 获取输出结果：

* 文本输出
* 文档预览
* 文档下载

#### 翻译服务接口层（Translate Service Interface Layer）

翻译服务接口层通过来自客户端的翻译请求，解析请求中的文本内容、源语言和目标语言等参数，然后将这些参数传递给后端的翻译引擎进行实际的翻译处理。处理完成后，接口层将翻译结果封装成HTTP响应并发送回Express应用层。在此过程中，接口层还需进行错误处理和安全性保障，如处理请求参数错误、翻译引擎故障等问题，并实施身份验证和数据加密等安全措施。总的来说，以上三个层级构成了一个完整的翻译软件系统，每个层级都负责不同的功能，各司其职。这 种分层架构模式使得系统更加清晰、易于扩展和维护。

### 界面原型设计

#### 主界面（Main Interface）

主界面是用户输入文本进行翻译的主要界面。

* **文本输入框**：位于页面中央，用户可以在此框内输入或粘贴需要翻译的文本。
* **翻译按钮**：紧邻文档上传组件边框，用户输入文本后点击此按钮即可开始翻译。
* **语言选择器**：位于页面顶部，用户可以在此选择源语言和目标语言。
* **翻译结果展示区**：位于文本输入框右边方，显示翻译后的文本结果。
* **复制按钮**：位于翻译结果展示区旁边和文本输入框两边，用户可以点击复制翻译后的文本和原文。

#### 文档上传界面（Document Upload Interface）

文档上传界面允许用户上传文档进行翻译。

* **文档选择按钮**：位于页面中央，用户可以点击此按钮选择要上传的文档。
* **上传文件列表**：显示已经上传过的文档。
* **语言选择器**：位于页面顶部，用户可以在此选择源语言和目标语言。
* **开始翻译按钮**：文档上传完成后，用户可以点击此按钮开始文档翻译。

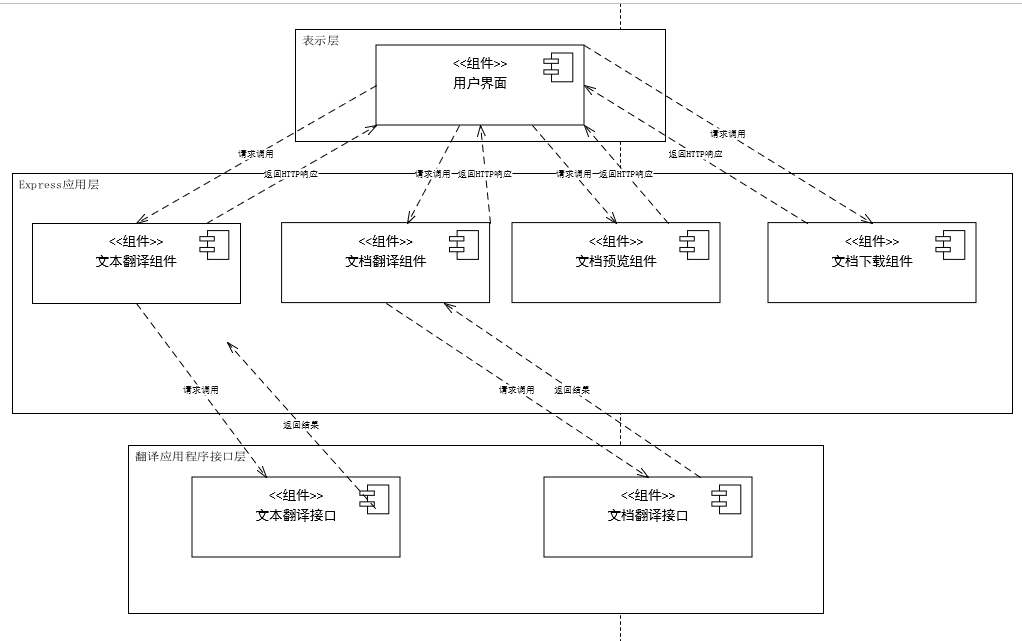
#### 文档预览界面（Document Preview Interface）

文档预览界面允许用户在翻译前预览文档内容。

* **源文档预览区**：位于页面左侧，显示上传文档的内容。
* **翻译文档预览区**: 位于页面右侧，显示上传文档的内容。
* **文档缩放按钮**:位于页面顶部，用于同步放大和缩小文档。
* **文档下载按钮**:位于页面顶部，用于下载翻译文档。

### 详细设计

#### 3.3.1. 组件设计



#### 3.3.2. 组件接口设计

##### 3.3.2.1.文档翻译接口

|  |
| --- |
|  |

* + - constructor(appid, seckey): 构造函数，初始化 appid 和 seckey。
    - generateRandomString(): 生成一个随机的字符串。
    - calculateStringMD5(inputString): 计算字符串的 MD5 值。
    - calculateFileMD5(filePath): 计算文件的 MD5 值。
    - uploadTranslate(from, to, format, filePath, edit): 上传文件进行翻译，参数包括源语言、目标语言、文件格式、文件路径和是否编辑文件名。
    - queryRes(tid): 查询翻译进度，参数为任务 ID。
    - download(tid): 下载翻译后的文件，参数为任务 ID。

##### 3.3.2.2Express应用接口设计

|  |
| --- |
|  |

* app.post('/upload', multipartMiddleware, function(req, res)): 处理文件上传请求。
* app.options('/upload', function(req, res)): 处理上传请求的预检请求。
* app.options('/translate', function(req, res)): 处理翻译请求的预检请求。
* app.post('/translate', async function(req, res)): 处理翻译请求。
* app.options('/queryRes', function(req, res)): 处理查询结果请求的预检请求。
* app.post('/queryRes', async function(req, res)): 处理查询翻译结果的请求。
* app.options('/download', function(req, res)): 处理下载请求的预检请求。
* app.post('/download', async function(req, res)): 处理下载翻译后文件的请求。
* app.get('/upload', function(req, res)): 获取上传状态。
* app.get('/downloadFile/:identifier', function(req, res)): 通过标识符下载文件。
* app.get('/download/:identifier', function(req, res)): 通过标识符下载文件。

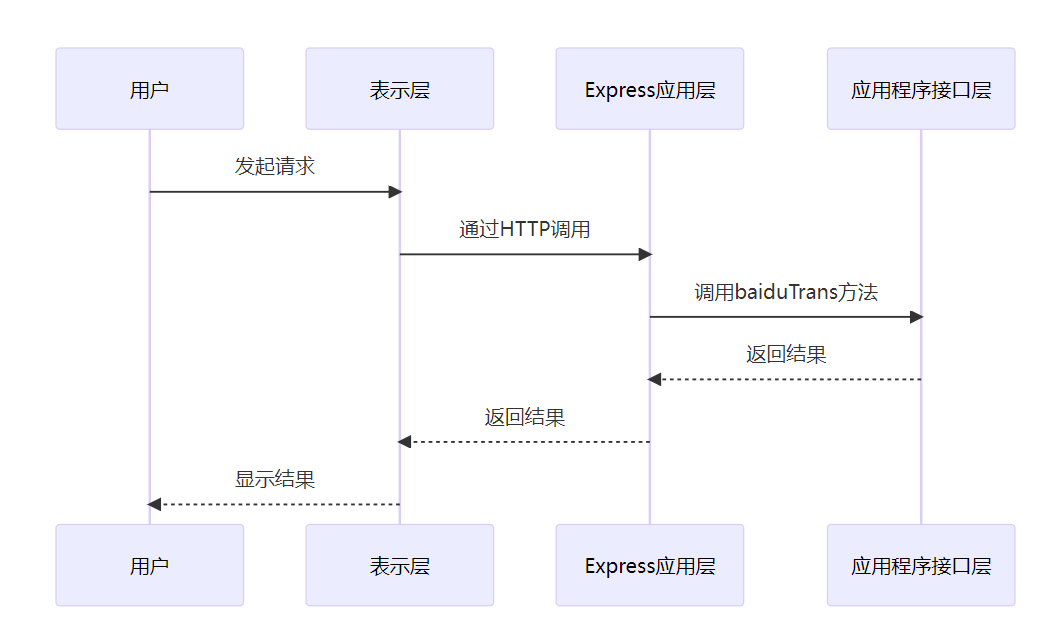
这些接口设计涵盖了文件上传、翻译处理、结果查询和文件下载等功能，为前端提供了清晰的调用方式和数据交互格式。

#### 3.3.3. 系统流程分析

|  |
| --- |
|  |

结合上述用例图，我们得出以下的时序图：

##### 1.文本翻译时序图



##### 2.文档上传时序图

|  |
| --- |
|  |

##### 3.文档翻译时序图

|  |
| --- |
|  |

##### 4.文档预览时序图

|  |
| --- |
|  |

##### 5.翻译文档下载时序图

|  |
| --- |
|  |