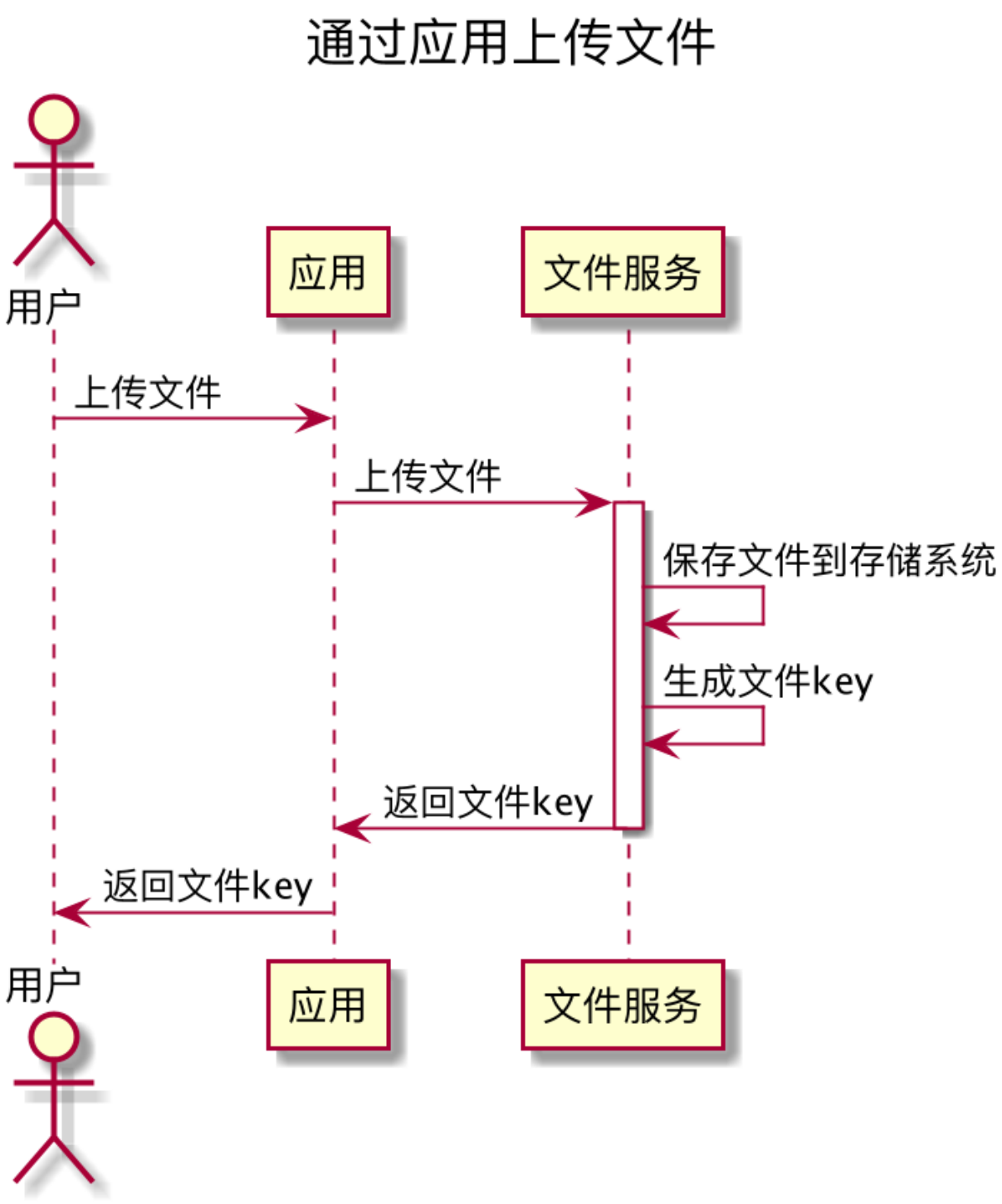
文件上传

由 夏玮创建, 最终由 夏玮修改于 把月 06, 2022

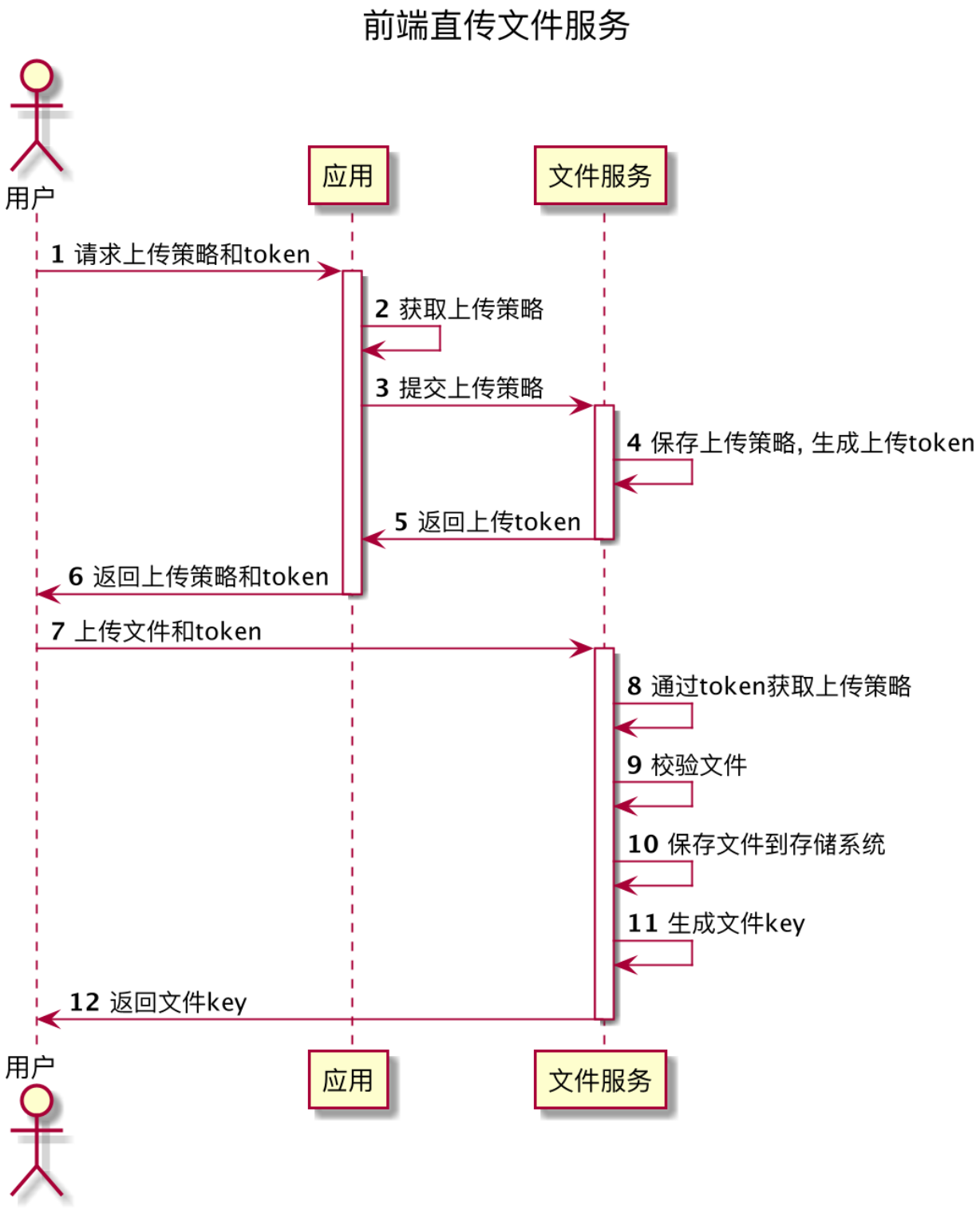
流程变更

通过应用上传文件



之前的文件上传是前端通过应用端上传到文件服务，等于经历了两次上传。如果前端能够直接上传到文件服务，可以避免不必要的传输，减少文件传输的损耗，减少对应用端性能的影响。

前端直传文件服务



对比通过应用上传，避免了应用传输到文件服务

使用方式

- pom中引入依赖

```
  <dependency>
    <groupId>cn.com.dbapp.adl</groupId>
    <artifactId>adl-common-core</artifactId>
    <version>0.0.1-SNAPSHOT</version>
  </dependency>
```

- 配置文件上传token生成入口 由于不同的业务场景对文件的限制不同，所以每个业务场景都需要定义自己的入口，避免上传的文件混乱。例如用户头像和题目附件是两个不同的业务场景，需要定义两个入口。

```
adl:
 file:
   upload:
     modules:
       test:
         sinatureToken: asdfasfasfdaf // 用来保证是否上传的文件是否是当前模块，不同模块一定要不同，否则文件限制可能被绕过
         limitSize: 10000 // 支持1000 10MB 10GB
         limitTypes: xls,txt
         url: /api/test // 上传token生成入口，前端通过该路径访问生成上传token
         expireSecond: 50 // 失效时间，单位 秒
```

- 前端上传文件到文件服务 前端选择文件后，带上应用生成的token，提交到文件服务，文件服务处理后返回key和signature，这两个参数需要前端在用户保存时，连同业务参数一起提交到应用端。
- 应用端接收前端提交的参数 后端使用固定的实体WebFileInfos去接收前端提交的参数，并在参数上增加@FileKeySignature(module = “test”, message = “{code}”)注解，module = "test"用来表明当前接收文件所属模块，而且需要保证有校验器校验。例如.

```
public class UserSaveDTO {
    @KeyFileSignature(module = "user-avatar", message = "{user.avatar.error}")
    private WebFileInfo avatar;
}

// 如果是Controller
public class UserController {

    public Response save(@Valid UserSaveDTO userSave) {
    }

}

// 如果是Service
public interface UserService {

    public Integer save(@Valid UserSaveDTO userSave);

}

@Validated
```

```
@Service
public interface UserServiceImpl implements UserService {

    @Override
    public Integer save(@Valid UserSaveDTO userSave) {
    }

}
```

- 文件更新 由于更新时需要给前端返回signature和url, 需要调用WebFileInfos中的fill(WebFileInfoFiller, String), 传入filler和module, filler会自动给WebFileInfos填充signature和url