基于博客网站的开发顺序

基于博客网站的目录结构，设计开发的顺序应遵循 **“从底层到上层、从核心到扩展”** 的逻辑，确保每一步都有可验证的成果，避免 “返工”。以下是分阶段的具体顺序，对应到前后端目录中的核心文件：

第一阶段：基础环境与项目骨架搭建（1-2 天）

目标：搭好 “空架子”，确保前后端能独立运行。

1. **后端初始化**
   - 用`django-admin startproject blog_backend`创建项目，生成`blog_backend/settings.py`、`manage.py`等核心文件。
   - 创建功能模块目录`apps/`，并在`settings.py`中注册（`INSTALLED_APPS`添加`'apps.article'`等）。
   - 配置数据库：在`settings.py`中设置 MySQL 连接（`DATABASES`配置），确保能连接数据库。
2. **前端初始化**
   - 用`vue create blog_frontend`创建项目，生成`src/main.js`、`public/index.html`等文件。
   - 新建核心目录：`src/views/`（页面）、`src/components/`（组件）、`src/api/`（接口）、`src/router/`（路由）。
   - 安装依赖：`npm install axios vue-router`（处理接口和路由），并在`main.js`中初始化路由和 Axios。

第二阶段：后端核心数据模型设计（1 天）

目标：定义 “数据长什么样”，为后续接口提供基础。

1. **设计并实现数据模型**
   - 优先实现`apps/article/models.py`（文章、分类模型）：定义`Article`（标题、内容、作者、时间等字段）、`Category`（分类名）。
   - 实现`apps/user/models.py`：如需扩展用户信息（如头像），可继承 Django 内置`User`模型；否则直接用内置`User`。
   - 实现`apps/comment/models.py`（评论模型）：定义`Comment`（内容、关联文章、关联用户、时间）。

2. **生成数据库表**
   - 执行`python manage.py makemigrations`生成迁移文件（`migrations/`目录下），再`python manage.py migrate`创建表。
   - （可选）创建超级用户`python manage.py createsuperuser`，通过 Django Admin（`/admin`）手动添加测试数据（如 2 篇文章、1 个分类）。

第三阶段：后端基础接口开发（2-3 天）

目标：实现 “数据增删改查” 的接口，供前端调用。

1. **文章相关接口**
   - 在`apps/article/serializers.py`中写序列化器：将`Article`模型转成 JSON（指定返回字段，如`id`、`title`、`content`）。
   - 在`apps/article/views.py`中写视图：用`ModelViewSet`实现文章的列表查询（`GET`）、单篇查询（`GET /id`）、发布（`POST`）、修改（`PUT`）、删除（`DELETE`）。
   - 在`apps/article/urls.py`中配置子路由：用`router.register('articles', ArticleViewSet)`注册接口，再在主`urls.py`中引入（`path('api/', include('apps.article.urls'))`）。
2. **用户相关接口**
   - 在`apps/user/views.py`中实现登录 / 注册接口（可用`django-rest-auth`简化开发，或自定义视图处理用户名 / 密码验证）。
   - 配置用户接口路由（`/api/login/`、`/api/register/`），并添加权限控制（如发布文章需登录）。
3. **测试接口**
   - 启动后端服务`python manage.py runserver`，用 Postman 或浏览器访问`http://127.0.0.1:8000/api/articles/`，确认能返回测试文章数据。

第四阶段：前端基础页面与路由（2 天）

目标：实现 “用户能看到的界面框架”，先静态后动态。

1. **配置路由** 使用TS而不是js 这里结合vue-router官方文档看下 `:id` 这种语法
   - 在`src/router/index.js`中定义路径：`/`→首页（`Home.vue`）、`/article/:id`→详情页（`ArticleDetail.vue`）、`/post`→发布页（`PostArticle.vue`）、`/login`→登录页（`Login.vue`）。
2. **实现基础页面**

- `src/views/Home.vue`：设计首页布局（导航栏 + 文章列表区域），先放静态示例（如 “测试文章 1” “测试文章 2”）。
- `src/views/ArticleDetail.vue`：设计详情页布局（标题 + 内容 + 评论区），静态展示示例内容。
- `src/components/Navbar.vue`：实现导航栏（包含首页、发布、登录按钮），引入到`App.vue`中让所有页面共用。

一般来说App.vue直接显示vue router的router-view
其他组件是在各自的分页 比如Home, About, Category等页面各自组装的

第五阶段：前后端联调（核心功能跑通）（2-3 天）

目标：让前端 “拿到后端数据” 并展示，实现基础交互。

1. **封装接口调用**
   - 在`src/api/article.js`中封装接口：

显然你需要在实现article.ts 之前先写一个reques

```
import request from '../utils/request'
export function getArticles() { return request.get('/api/articles/') } // 获取文章列表
export function getArticle(id) { return request.get(`/api/articles/${id}/`) } // 获取单篇文章
export function postArticle(data) { return request.post('/api/articles/', data) } // 发布文章
```

2. **首页展示文章列表**
   - 在`Home.vue`中调用`getArticles()`，用`v-for`循环渲染后端返回的文章数据（替换静态示例）。
3. **详情页展示单篇文章**
   - 在`ArticleDetail.vue`中通过路由参数`this.$route.params.id`获取文章 ID，调用`getArticle(id)`，渲染标题和内容。
4. **解决跨域问题**

跨域问题做好笔记，仔细研究，和浏览器的安全策略有关，需要仔细了解浏览器的安全策略是怎么回事，为什么这么设计，单独写一篇笔记

   - 后端安装`django-cors-headers`，在`settings.py`中配置`CORS_ALLOWED_ORIGINS = ["http://localhost:8080"]`（允许前端域名访问）。

第六阶段：进阶功能开发（3-5 天）

目标：完善用户交互和业务逻辑。

(你真的需要登陆注册功能吗)

1. **用户登录 / 注册**
   - 前端`Login.vue`实现登录表单，调用`src/api/user.js`中的登录接口，成功后将 token 存入`localStorage`。

- 后端接口添加权限验证（如`ArticleViewSet`中用`IsAuthenticated`限制发布文章需登录）。

2. **发布 / 编辑文章**
   - 前端`PostArticle.vue`实现富文本编辑器（可选`vue-quill-editor`），表单提交时调用`postArticle(data)`，并带上登录 token。
3. **评论功能**
   - 后端`apps/comment/`实现评论接口（提交、查询），关联文章和用户。
   - 前端`ArticleDetail.vue`中添加评论表单，调用评论接口，并用`CommentList.vue`组件展示评论列表。
4. **分类筛选**
   - 后端`ArticleViewSet`添加分类过滤（`filterset_fields = ['category']`），前端点击分类时调用`/api/articles/?category=1`筛选文章。

第七阶段：测试与优化（1-2 天）

测试优化可以先跳过，先部署了最后再优化

1. **功能测试**：验证所有流程（注册→登录→发布→展示→评论）是否正常，修复 bug（如未登录用户无法发布）。
2. **前端优化**：
   - 给`ArticleCard.vue`添加懒加载，优化图片加载速度。
   - 用`v-if`控制未登录用户隐藏 “发布” 按钮。
3. **后端优化**：
   - 给`Article`表的`title`和`create_time`加索引，加速查询。
   - 限制接口请求频率（用`django-ratelimit`防刷）。

第八阶段：部署上线（1 天）

MySQL部署比较麻烦，如果最后有问题还是可以换回sqlit
django可以无缝切换不同的关系数据库 这是ORM的优点

1. **后端部署**：
   - 服务器安装 Python、MySQL，上传代码，安装`requirements.txt`依赖。
   - 用`gunicorn`启动服务，`Nginx`反向代理（处理静态文件和域名绑定）。
2. **前端部署**：
   - 执行`npm run build`生成`dist`目录，上传到服务器，用`Nginx`部署静态文件。
3. **域名与 SSL**：绑定域名，申请 SSL 证书（让网站支持 https）。

可以不用SSL。到时候再说

按这个顺序开发，每阶段都能看到明确的成果（如 “阶段 3 后能访问接口”“阶段 5 后首页能显示真实文章”），适合新手循序渐进。如果时间紧张，可优先完成 “阶段 1-5”（核心功能：看文章、发文章），后续再补全评论、分类等功能。

> （注：文档部分内容可能由 AI 生成）