# Git 代码规范

## 一、分支命名

1. master：主分支，用于部署生产环境，保持稳定性，一般由 release 及 hotfix 分支合并，不可直接修改代码。

2. develop：开发环境分支，始终保持最新完成及 bug 修复后的代码，用于前后端联调。

3. feature：开发新功能分支，以 develop 为基础创建，命名如 feature/user\_module、feature/cart\_module。

4. test：测试环境分支，外部用户无法访问，给测试人员使用，版本相对稳定。

5. release：预上线分支，预发布阶段使用，一般由 test 或 hotfix 分支合并，不建议直接在 release 分支修改代码。

6. hotfix：线上紧急问题修复分支，以 master 分支为基线，修复后合并到 master 和 develop 分支。

## 二、分支与环境对应关系

- DEV：开发者调试，对应 develop 分支。

- FAT：功能验收测试环境，对应 test 分支。

- UAT：用户验收测试，对应 release 分支。

- PRO：生产环境，对应 master 分支。

- 表格说明：

| 分支 | 功能 | 环境 | 可访问 |

|---|---|---|---|

| master | 主分支，稳定版本 | PRO | 是 |

| develop | 开发分支，最新版本 | DEV | 是 |

| feature | 开发分支，实现新特性 | - | 否 |

| test | 测试分支，功能测试 | FAT | 是 |

| release | 预上线分支，发布新版本 | UAT | 是 |

| hotfix | 紧急修复分支，修复线上 bug | - | 否 |

## 三、提交类型规范

- feat：新增功能

- fix：修复 bug

- docs：仅文档更改

- style：不影响代码含义的更改（空白、格式设置、缺失分号等）

- refactor：既不修复 bug 也不添加特性的代码更改

- perf：改进性能的代码更改

- test：添加缺少的测试或更正现有测试

- chore：对构建过程或辅助工具和库（如文档）的更改

## 四、单次提交注意事项

1. 提交问题必须为同一类别。

2. 提交问题不要超过 3 个。

3. 提交的 commit 发现不符合规范，使用 git commit --amend -m "新的提交信息" 或 git reset --hard HEAD 重新提交一次。