ChatGPT

# Taiga 开源项目最新版本调研报告

## Taiga 项目概述

**Taiga 是什么？** Taiga 是一个免费开源、功能强大的项目管理平台，专为初创企业和敏捷开发团队设计 [1]。它注重简洁和可定制性，支持敏捷开发中的 **Scrum** 和 **Kanban** 方法论 [2]。Taiga 提供了丰富的项目管理功能，核心功能包括直观的看板、待办事项列表、Wiki 知识库和问题跟踪等 [3]。通过这些功能，团队可以轻松规划冲刺和任务，跟踪项目进度，并实现团队协作与知识共享。例如，Taiga 内置的看板功能让团队以看板方式管理任务状态，Wiki 模块用于文档协作，问题跟踪模块则用于记录和处理缺陷或工作项 [3]。总之，Taiga 作为敏捷项目管理工具，可帮助团队高效地协同工作、透明地跟踪项目进展，并支持与版本控制系统等外部工具集成（如 Git 集成实现代码提交与任务状态同步，以及 Webhooks 触发自动流程） [4]。

**Taiga 的核心用途和特点：**

- **敏捷项目管理：** 支持 Scrum 和 Kanban 两种主要敏捷开发流程，满足不同团队的流程需求 [2]。用户可以使用待办列表规划产品 backlog，用迭代（冲刺）管理短期任务。
- **任务与缺陷跟踪：** 提供任务卡片来跟踪每个工作项的状态、优先级、标签等；内置缺陷/问题跟踪系统，确保团队及时发现并解决项目问题 [5]。
- **知识共享与文档：** 内置 Wiki 和文档模块，方便团队编写项目说明、需求文档和知识库，集中管理项目相关知识 [6]。
- **角色权限与通知：** 支持自定义用户角色和权限控制，不同角色可访问不同范围的项目内容；提供邮件及系统内通知功能，及时提醒相关成员关注项目更新 [7]。
- **扩展与集成：** 拥有开放的 REST API，可与其他工具集成 [8]；支持 Git 等版本控制系统无缝集成，提交代码可自动关联任务状态 [9]；支持 Webhooks，用于触发自动化工作流程 [7]。

以上特点使得 Taiga 成为中小型团队理想的项目管理方案，兼具**易用的界面**和**灵活的定制性**，能够满足团队在敏捷开发过程中的协作和管理需求 [1]。

## Taiga 最新版本的主要仓库和模块

最新版本的 Taiga 平台由多个独立的模块组成，每个模块对应一个 GitHub 仓库 [10]。这些模块分工明确，通过 API 调用、消息队列和共享配置等方式协同工作，共同构成完整的 Taiga 平台。下表概述了 Taiga 核心仓库的功能定位，以及它们之间的依赖关系和交互逻辑：

| 仓库名称 | 功能与作用 | 依赖关系与交互逻辑 |
| --- | --- | --- |
| **taiga-back** | Taiga 后端（Backend/API）。提供主要的业务逻辑和 REST API 服务，是 Taiga 的核心后端组件 [11]。后端基于 Python 的 **Django** 框架开发，并使用 **PostgreSQL** 作为主要数据库 [11]。 | **依赖/数据源：** 依赖 PostgreSQL 数据库存储项目数据；可选地依赖 **RabbitMQ** 消息队列实现异步任务和实时消息推送 [12]。<br>**对外交互：** 向前端 (taiga-front) 暴露 REST API 接口，供前端获取和修改项目数据 [11]。当开启异步模式时，taiga-back 会通过 RabbitMQ 将耗时任务交给异步任务队列处理（即 taiga-async，运行 Celery 分布式任务机制） [12]。此外，taiga-back 在有状态更新时也会向 RabbitMQ 发布事件消息，以便 **taiga-events** 服务器获取并通知前端，实现实时更新 [13]。taiga-back 同时包含附件保护插件 (taiga-contrib-protected) 的后端实现，用于与 **taiga-protected** 服务协作保护附件访问 [14]。 |
| **taiga-front** | Taiga 前端（Frontend）。这是 Taiga 的用户界面层，以单页 Web 应用形式呈现 [15]。Taiga 前端提供项目的网页界面，包括看板视图、任务列表、问题列表、Wiki 页面等交互界面，供最终用户使用。前端代码使用 JavaScript 框架开发（Taiga 6 版本使用 **AngularJS** + CoffeeScript） [16]。 | **依赖/数据源：** 依赖 taiga-back 提供的 API 数据。前端启动时需要加载一个配置文件（`conf.json`），其中指定后端 API 的 URL 及 WebSocket 服务器地址等 [17]。<br>**对外交互：** 前端通过 HTTP 请求调用 **taiga-back** 的 REST API 来读取或修改项目数据（例如获取项目列表、更新任务状态等） [11]。同时，前端通过 WebSocket 连接 **taiga-events** 服务，以接收实时事件通知，实现看板和任务列表的实时刷新 [13]。用户在前端进行的操作（如新建任务、修改状态）会通过 API 提交到后端处理，后端更新数据后发送事件消息通知前端界面。 |
| **taiga-front-dist** | Taiga 前端发布版。该仓库包含编译打包后的前端静态文件，即 **taiga-front** 源代码构建出的发布版本 [18]。它用于**生产环境部署**，方便直接提供前端页面，而无需在部署时重新编译源码。taiga-front-dist 的内容通常包括压缩后的 HTML、JavaScript、CSS 以及前端运行所需的配置文件。 | **依赖/数据源：** 前端发布版在运行时仍然依赖 taiga-back 的 API 和 taiga-events 的 WebSocket 服务提供数据与实时功能，但自身不包含动态后端逻辑。<br>**对外交互：** **taiga-front-dist** 通常通过 Web 服务器（如 **Nginx**）以静态文件形式对外提供服务 [19]。部署时，Nginx 会将对 `/` 根路径的访问指向 taiga-front-dist 的静态文件（即前端应用），将对 `/api` 路径的请求反向代理给 taiga-back 接口，将 `/events` 的 WebSocket 请求代理给 taiga-events 服务等 [20] [21]。因此，taiga-front-dist 本身不直接“交互”其他模块，而是通过浏览器加载后，与后端和事件服务器交互（这一过程与开发环境下的 taiga-front 一致）。 |

| 仓库名称 | 功能与作用 | 依赖关系与交互逻辑 |
| --- | --- | --- |
| **taiga-events** | Taiga 实时事件服务器。taiga-events 是一个 **WebSocket 网关** 服务，用于实现 Taiga 平台的实时功能 [13]。当项目中发生状态变更（如新建任务、修改任务状态等）时，此组件负责将变更通知推送给连接的前端客户端，使多个用户的界面同步更新。taiga-events 基于 **Node.js** 开发，并利用 **RabbitMQ** 作为消息中间件 [13]。 | **依赖/数据源：** 依赖 RabbitMQ 消息队列从 taiga-back 接收事件消息 [13]。也依赖 taiga-back 提供的一个共享密钥（用于安全验证 WebSocket 通信，可在 taiga-back 配置中设置 events 模块密钥）。<br>**对外交互：** **taiga-events** 通过 RabbitMQ 订阅 taiga-back 发布的事件，当检测到有新的事件消息时，会通过 WebSocket 将更新发送给所有连接的前端客户端 [13]。前端（taiga-front 或 taiga-front-dist）在加载时会建立与 taiga-events 的 WebSocket 连接（地址通常为 `ws://<server>/events`），从而实时接收任务、用户故事等数据的变更通知，实现多人协作时界面的实时同步。 |
| **taiga-protected** | Taiga 附件保护服务。taiga-protected 模块是一个独立的微服务，用于保护项目附件（如上传的图片、文档等）免于未经授权的直接访问 [14]。Taiga 中的附件保护机制通过**签名 URL/令牌**实现：当用户请求下载受保护的附件时，需要一个临时有效的访问令牌。taiga-protected 服务负责验证该令牌的有效性，从而决定是否允许访问附件内容 [14]。 | **依赖/数据源：** 依赖 taiga-back 中安装的 **taiga-contrib-protected** 插件配合使用。taiga-back 在返回附件下载链接时，会生成包含签名令牌的 URL；该令牌由 taiga-back 使用与 taiga-protected 共享的密钥加密生成 [22]。taiga-protected 使用相同的密钥配置来验证令牌。还依赖于 Nginx 或其他 Web 服务器的配置配合：Nginx 拦截对受保护附件路径的请求（通常是在 `/protected` 或特定路径下），并将请求转发给 taiga-protected 进行校验 [14]。<br>**对外交互：** **taiga-protected** 对 Nginx 充当**内部授权网关**：当用户下载某附件时，Nginx不会直接提供文件，而是将请求交给 taiga-protected 服务。taiga-protected 校验请求中的令牌，如果通过则让 Nginx 放行并从附件存储目录提供文件，否则拒绝请求（或返回未授权）。通过这种机制，实现只有授权用户才能访问私有项目附件的目的 [14]。 |
| **taiga-docker** | Taiga 的 Docker 部署仓库。该仓库包含一系列 Docker 镜像和 **Docker-Compose** 配置，旨在使 Taiga 在容器环境中快速部署和运行 [23]。taiga-docker 提供了所需的配置文件（如 `.env` 环境变量文件、`docker-compose.yml` 编排文件，以及 `taiga.conf` Nginx 配置等）来 orchestrate Taiga 的各个服务。通过 taiga-docker，用户可以一键启动包括 **taiga-back**, **taiga-front-dist**, **taiga-events**, **taiga-protected** 以及依赖服务（如 PostgreSQL 数据库、RabbitMQ 消息队列、Redis、Nginx 网关等）在内的多个容器。 | **依赖/数据源：** 该仓库不包含业务逻辑代码，但依赖各 Taiga 镜像（taiga-back 等）的发布版本。它提供的 `.env` 文件允许配置各组件的环境变量（如数据库密码、域名、端口等） [23]。同时包含标准的 **Nginx** 配置（taiga-gateway）作为入口，将不同的 URL 路径分别代理到对应的 Taiga 服务容器上 [19]。<br>**对外交互：** **taiga-docker** 本身作为部署工具，没有运行时交互功能。它定义的 Nginx 服务（taiga-gateway 容器）对外提供统一入口：例如，静态前端由 Nginx 提供，`/api` 请求由 Nginx 转发给 taiga-back 容器的 API 接口，`/events` WebSocket 请求转发给 taiga-events 容器，`/media` 附件请求根据受保护与否转发至 taiga-back 或 taiga-protected 等 [21]。通过 docker-compose 定义，taiga-docker 协调各容器的启动顺序、网络联通和数据卷共享，使各模块能够在容器内正确协作。 |

**表：Taiga 核心 GitHub 仓库及其功能、交互概览（基于最新版本）。**

上表列出了 Taiga 平台的主要组成模块和对应的 GitHub 代码仓库，以及它们的职责和相互关系。其中 taiga-back、taiga-front(-dist) 和 taiga-events 是 **Taiga 平台的核心三大组件** [11]；taiga-protected 和异步任务 (taiga-async) 则属于可选的附加组件，用于增强附件安全和提高性能等 [24]。taiga-docker 则是官方提供的容器化部署方案仓库，方便用户将上述组件快速部署在 Docker 环境中。下面将对这些仓库之间的依赖关系和交互原理作进一步说明。

## 仓库之间的依赖关系与架构原理

Taiga 采用前后端分离的模块化架构，各组件通过明确的接口进行通信，整体架构如下：

- **前后端通信：** Taiga 的 Web 前端（taiga-front 或 taiga-front-dist）通过调用后端 API 来实现大部分功能。用户在界面上的操作（如创建用户故事、修改任务状态等）会以 REST 请求形式发送到 **taiga-back**，后端处理业务逻辑和数据库读写，然后返回结果给前端 [11]。这种松耦合的 API 通信使前后端可以独立开发和部署。

- **实时更新机制：** 为了在多人协作时提供实时反馈，Taiga 引入了 **taiga-events** WebSocket 服务 [13]。当后端数据发生变更时（例如某人完成了一项任务），**taiga-back** 会向 RabbitMQ 发布一条事件消息 [13]。**taiga-events** 作为订阅者捕获到该消息后，通过 WebSocket 将更新推送给所有连接的前端客户端，使他们的界面**实时刷新**相应的内容 [13]。例如，在任务看板上，一位用户拖动卡片改变了任务状态，其他用户的看板上几乎瞬时就会反映出该变化。RabbitMQ 在此承担了解耦发布/订阅的角色，而 taiga-events 则是后端和前端之间的**实时消息桥梁**。

- **异步任务处理：** **taiga-back** 默认情况下同步执行所有操作，但对于发送邮件、导入导出等耗时任务，可以启用 Taiga 的异步处理模式（即 **taiga-async**）[12]。在该模式下，Taiga 后端会将耗时工作投递到 RabbitMQ 队列，由独立的 Celery Worker 进程消费执行 [12]。这个 Celery Worker 可以理解为 taiga-back 代码的另一个运行实例，专门用于执行异步任务（通常通过 Docker Compose 定义为一个独立服务，如 `taiga-async` 服务使用 taiga-back 镜像启动 Celery）。这样可以避免阻塞主应用，提高系统吞吐量。不过值得注意的是，**taiga-async** 并没有独立的代码仓库，其逻辑实现包含在 taiga-back 中，只是通过配置决定是否以异步模式运行而已 [12]。RabbitMQ 同时被 taiga-events 和 taiga-async 利用，分别服务于实时通知和异步任务两个机制。

- **附件保护机制：** 对于私有项目中的附件文件，Taiga 提供了可选的 **taiga-protected** 模块进行保护 [14]。在没有 taiga-protected 时，附件通常由后端直接提供，或者通过静态服务器开放，这可能导致未经授权的访问。引入 taiga-protected 后，Taiga 后端会对附件 URL 进行签名，只有携带有效令牌的请求才能通过 Nginx 获取实际文件 [14]。部署时，Nginx（taiga-gateway）配置了一个内部路径（例如 `/protected`）由 taiga-protected 服务处理 [25]。当用户尝试下载某附件时，如果该附件受保护，前端会请求一个含签名的下载链接；浏览器据此向 Nginx 发起下载，Nginx 将请求转给 taiga-protected 验证令牌 [14]。验证通过后，Nginx 才从存储卷中读取实际文件（或通过后端提供文件）返回给用户，否则拒绝访问。令牌通常设置短期限（例如几分钟），过期后需重新请求，以确保附件链接无法长期传播 [26]。这一机制确保了**附件资源只能被授权用户访问**，强化了 Taiga 在安全性方面的能力。

- **统一网关与部署：** 在生产部署中，通常使用 Nginx 作为前端网关服务器，它通过反向代理将请求分别导向不同的 Taiga 服务。Taiga 官方的 Docker 部署(**taiga-docker**)仓库提供了示例 Nginx 配置 `taiga.conf` [19]。根据该配置，Nginx 将：`/` 根路径下的请求交给前端容器（taiga-front-dist）处理静态页面，【`/api`】前缀的请求转发给 taiga-back 容器的 API，【`/events`】路径升级为 WebSocket 并代理到 taiga-events，【`/media`】和【`/static`】等静态资源路径映射到后端或静态卷（部分受 taiga-protected 控制）[21]。这种架构下，Nginx 扮演了**集中入口**的角色，客户端只需要访问 Nginx 的域名和端口即可使用 Taiga 的全部功能，由 Nginx 来协调后端各服务。taiga-docker 则利用

Docker Compose 将上述所有组件和依赖服务（数据库 Postgres、消息队列 RabbitMQ、缓存 Redis 等）定义在一起，一次性部署。根据官方文档说明，一个标准的 Taiga 部署包含上述 **taiga-back、taiga-front-dist、taiga-events、taiga-async 和 taiga-protected** 等多个模块，它们既可以部署在同一台机器上，也可以分布部署 [27]。使用 taiga-docker 则默认将它们部署为相互连接的容器，以简化搭建流程 [23]。

综上，Taiga 的架构体现了典型的**前后端分离 + 异步消息 + 微服务**设计思想。各仓库模块通过明确的接口（HTTP API、WebSocket、消息队列、共享配置等）集成，实现了**功能解耦**和**水平扩展**：前端界面与后端逻辑解耦，实时通知与主流程解耦，附件服务独立，部署上也可按需拆分。这使得 Taiga 在保证丰富功能的同时，仍能保持较好的可维护性和扩展性。

## 官方文档与架构资料

关于 Taiga 各组件和仓库关系的更多信息，可以参考官方提供的文档和社区资源：

- **官方文档：** Taiga 官方文档的安装和开发指南部分对平台架构有基本说明。例如，开发环境设置指南中指出 “Taiga 平台由三个主要组件组成：taiga-back（后端API）、taiga-front（前端）和 taiga-events（WebSocket 实时网关）”，并提到 taiga-events 是可选组件 [10]。生产部署文档则进一步列出了标准 Taiga 平台包括 taiga-front-dist 前端发布版、taiga-async 异步任务和 taiga-protected 附件保护等模块 [27]。这些官方文档有助于理解各仓库模块的职责划分和部署拓扑。

- **架构示意与讨论：** 在 Taiga 社区论坛，有开发团队成员对 Taiga 的架构进行了总结说明。例如，有帖子引用了上述官方文档内容，并补充说明了 taiga-async 和 taiga-protected 等可选模块的作用 [24]。这些讨论提及了 Taiga 技术栈（Django、AngularJS、RabbitMQ、Celery等）以及各部分协同工作的方式，对理解仓库之间的交互很有帮助。

- **仓库自述文件：** 大部分 Taiga 仓库（如 taiga-back、taiga-front、taiga-events 等）的 README 中也提供了简要的介绍和指向更详尽文档的链接。例如，taiga-back 和 taiga-front 的 README 提供了文档网址和社区支持链接 [28] [29]。taiga-front-dist、taiga-events、taiga-protected 等仓库的 README 冒头通常有提示，指向官方博客公告（如 Taiga 未来规划）以及相关文档 [18] [30]。阅读这些说明有助于快速了解仓库用途，以及获取进一步配置指导（例如 taiga-protected README 提供了插件安装和 Nginx 配置的文档链接 [14]）。

  **资料链接：** 如果需要，可参阅 Taiga 官方文档站点的 安装部署指南 和 开发者指南，以及 Taiga 社区的相关讨论帖获取更深入的架构说明 [27] [24]。上述资源详细阐述了各模块的安装配置和作用原理，对本次调研内容提供了权威参考。

---

[1] [2] [4] [5] [6] [7] [8] [9] [19] [23] 开源项目管理工具Taiga-CSDN博客
https://blog.csdn.net/wbsu2004/article/details/141642374

[3] 如何利用Taiga项目管理工具提升团队效率？5个实用技巧分享 - 老杨的PM手记 - SegmentFault 思否
https://segmentfault.com/a/1190000047025465

[10] Taiga: Setup development environment
https://docs.taiga.io/setup-development.html

[11] [12] [13] [15] [24] What technologies are being used in Taiga? - Self hosted Taiga - Taiga Community
https://community.taiga.io/t/what-technologies-are-being-used-in-taiga/2358

[14] [22] [25] [30] GitHub - taigaio/taiga-protected
https://github.com/taigaio/taiga-protected

[16] How to Install Taiga.io Project Management Software on CentOS 7
https://www.howtoforge.com/tutorial/how-to-install-and-configure-taigaio-on-centos-7/

[17] [26] [27] Install Taiga in Production
https://docs.taiga.io/setup-production.html

[18] [29] GitHub - taigaio/taiga-front-dist
https://github.com/taigaio/taiga-front-dist

[20] [21] Synology 部署开源项目管理工具 Taiga | 飘零博客
https://www.liangz.org/1143.html

[28] GitHub - taigaio/taiga-back
https://github.com/taigaio/taiga-back