文档编号：Minellius– SRS – 0.3.1

Minellius

软件需求规格说明书

名字	学号	贡献描述
周烨恒	SZ160110103	初稿，第 3、4、5 章，规格说明错误修复
方安	SZ160110101	第 1、2 章
林锦涛	SZ160110203	第二版及第三版修正
崔天宇	SZ160510117	第二版及第三版修正
曹巍瀚	SZ160610219	第二版及第三版修正

文档变更历史记录

序号	变更日期	变更人员	变更内容详情描述	变更后的版本号
1	2018 年 11 月 26 日	周烨恒，方安	创建文档	0.0.1
2	2018 年 11 月 28 日	方安	项目概述、引言	0.0.2
3	2018 年 12 月 1 日	方安	一般性描述和其它需求描述	0.0.3
4	2018 年 12 月 2 日	周烨恒	完整填充，初版可用需求书发布	0.1.0
5	2018 年 12 月 6 日	周烨恒	丰富内容，错误和细节修正，添加参考	0.1.1
6	2018 年 12 月 20 日	崔天宇、林锦涛、曹巍瀚	修改了软件原型示例图	0.2.1
7	2018 年 12 月 27 日	周烨恒、崔天宇、林锦涛、曹巍瀚	修改了数据库说明	0.3.1

一. 引言

1.1 编写目的

1. 为开发小组成员、客户之间提供共同的协议而创立基础；
2. 对Minellius的主要功能、性能进行完整描述；
3. 安排项目规划与进度、组织软件开发与测试。

1.2 读者对象

本说明书的预期读者为开发小组成员及甲方负责人。

1.3 软件项目概述

1.3.1 项目名称

`Minellius`

1.3.2 用户单位

面向全社会，开放使用。对于软件本身，可以在 `MIT LICENSE` 许可证限制的范围内加以修改、分发和使用；对于相关文本性质文档资料的使用，需要遵守 `Common Creative By 4.0` 许可。

1.3.3 开发单位

哈尔滨工业大学（深圳） `Minellius` 开发小组

1.3.4 大致功能和用途等

- `Minellius` 是一套实验性的GITHUB数据解决方案，它将集数据收集、持久化、整合、分析、可视化、实时监控、趋势追踪及便利设施于一体，提供新型、全方位的服务。
- 它旨在帮助用户了解开源世界的历史、当下、潮流和沉浮兴衰，提升时代嗅觉和社区文化审美；调整开发者学习、自我提升的路径，以更好的融入、参与社区资源文化的发展和建设。

1.4 文档概述

文档分为六个章节：

1. 将从产品软件与其环境之间的环境以及限制与约束、假设与前提条件这三个方面对软件进行一般性描述。
2. 在功能需求章节中，将系统划分为若干相关子系统，并通过用例图和描述详细介绍这些子系统的功能。
3. 同时，将从性能要求、设计约束、界面要求、进度要求、交付要求和验收要求这六个方面的要求，进一步对软件及其其开发过程进行详细描述。

1.5 定义

【术语1】：数据流图

说明：数据流图是结构化方法中用于表示系统逻辑模型的一种工具。它将描述系统由哪几部分组成，各部分之间有什么联系等；它以图形的方式描绘数据在系统中流程和处理的过程。DFD只反映系统必须完成的逻辑功能。

【术语2】：用例图

说明：用例图是指由参与者、用例，边界以及它们之间的关系构成的用于描述系统功能的视图。用例图是外部用户所能观察到的系统功能的模型图。用例图是系统的蓝图。用例图呈现了一些参与者，一些用例，以及它们之间的关系，主要用于对系统、子系统或类的功能行为进行建模。

1.6 参考资料

1. Software Engineering: A Practioner's Approach, Eight Edition.
2. Guide to Software Requirements Specifications，IEEE.
3. Ant Design, Alibaba.

二. 软件的一般性描述

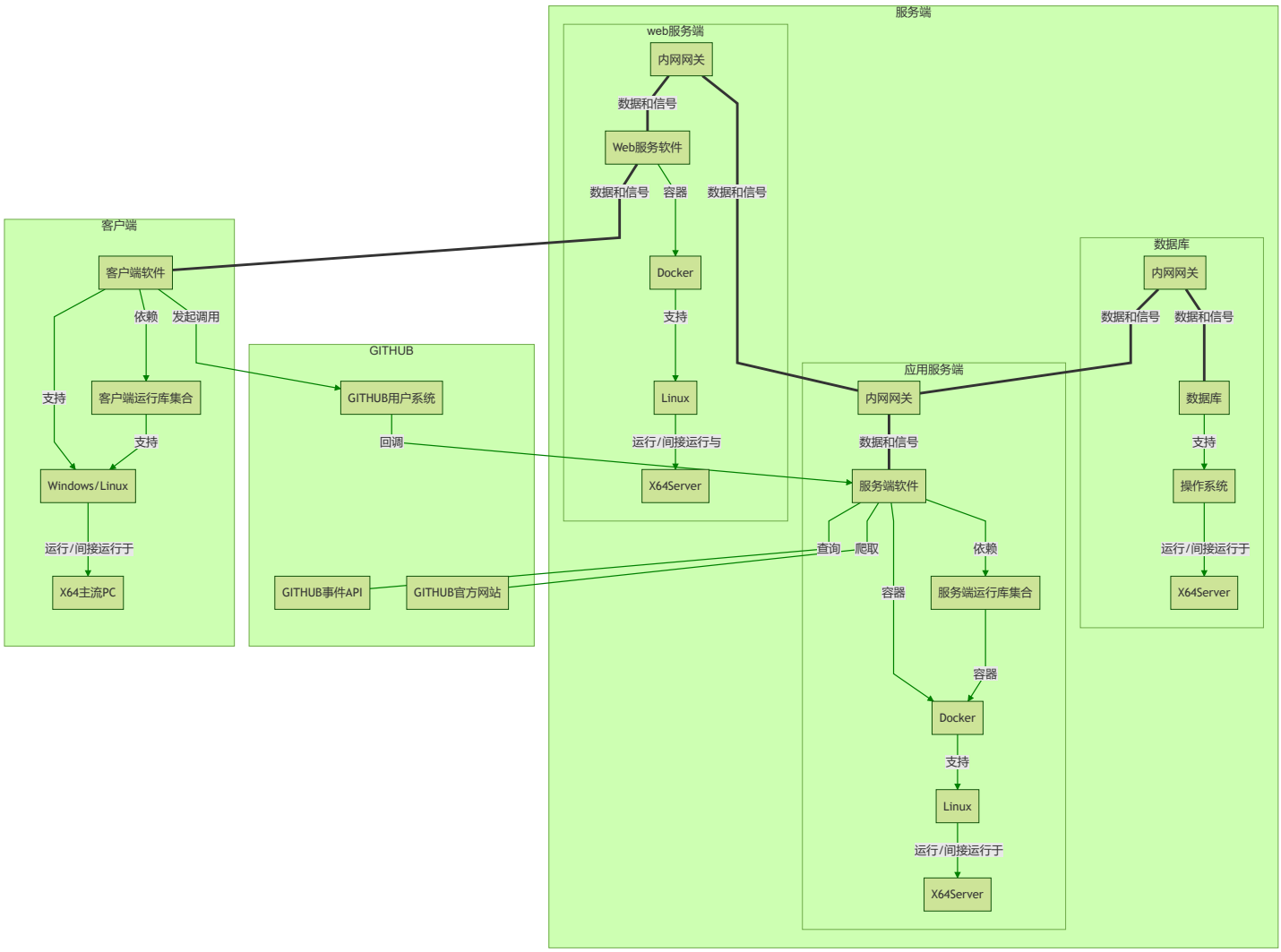
2.1 软件产品与其环境之间的关系

2.1.1 外部的用户

准 顶层数据流图

2.1.2 外部的系统

准 外部系统关系图



2.2 限制与约束

2.2.1 硬件的限制

截止当前版本，仅限于在 `X64` 硬件架构平台上的 `Linux`、`Windows` 使用对应版本的客户端。

细节性的硬件限制见 4.1 其它需求描述

2.2.2 与其他应用的接口

与社交媒体等用户系统的对接

暂定仅给出与 github 用户系统的对接，通过再官方注册应用，以 `OAuth 2.0` 协议进行对接，并通过回调获取返回的结果。

与后端网关的接口

由 `nginx/openresty` 为应用提供反向代理和均衡负载，通过 HTTP 系列协议。

2.2.3 所需的高级语言

服务端语言

- `python`：历史增量数据爬取和处理
- `typescript`：除前者外所有功能

客户端语言

- `typescript`：动态渲染和功能
- `html`：页面模板
- `css/scss/postcss`：样式

2.2.4 通信协议

网络层：`IP`

传输层：`TCP`

应用层

`HTTP` - 最低 `HTTP/1.1`，兼容 `HTTP/2`
`TLS` - `TLS1.2`

2.2.5 安全和保密方面的考

通讯安全

- 协议：基于 `TLS1.2` 的 `HTTPS`
- 验证：`Let's Encrypt` 的 `acme.sh`
- 方式：双栈
    1. `ECC`
        - 加密位数：`256 bit`

        - 技术细节：`TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256`
    2. `RSA`
        - 加密位数：`2048 bit`
        - 技术细节：`TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256`
- `HTTP/2`：支持
- `OCSP`：支持
- 预防降级攻击：支持
- 正向保密：支持
- `HTTP` 严格传输安全：支持（`max-age=315360000`）
- 会话恢复：`caching`，`ticket`
- 服务端安全重协商：支持
- 客户端安全重协商：支持
- `HTTP` 访问控制（`CORS`）：支持

认证安全

- 认证架构：`OAuth 2.0`
- 验证方案：`bearer` 令牌
- 认证传输：`json web token`

数据安全

- 密码存储：`salted`
- 第三方数据调用：`OAuth 2.0` 回调，不涉及敏感信息泄露

2.3 假设与前提条件

2.3.1 影响需求的的假设因素：

使用的第三方和商业组件

该项目发布许可证为 `MIT LICENSE`，使用的第三方组件皆为以兼容协议/许可证发布，文档等文本内容发布协议遵循 `Common Creative By 4.0`：

组件	许可证	兼容性
mariadb	GPL	网络端口的许可证隔离
nginx/openresty	BSD-2-Clause	许可证兼容
nest*	MIT	许可证兼容
node*	MIT	许可证兼容
typescript*	APACHE 2.0	许可证兼容
python*	PYTHON	许可证兼容

组件	许可证	兼容性
angular*	MIT	许可证兼容
lodash	MIT	许可证兼容
rxjs	APACHE 2.0	许可证兼容
ng-zorro-antd	MIT	许可证兼容
electron	MIT	许可证兼容
crawler	MIT	许可证兼容
yarn	BSD-2-Clause	许可证兼容
jest*	MIT	许可证兼容
webpack*	MIT	许可证兼容
babel*	MIT	许可证兼容
passport*	MIT	许可证兼容
pandas	BSD-3-Clause	许可证兼容
其它直接依赖	多种许可证	许可证兼容
间接依赖	多种许可证	由对应直接依赖兼容

`*` 表示系列工具，它们拥有相同或者兼容的许可证，后方所示许可证为主体项目许可证。

操作和开发环境的问题约束

1. 操作环境：
    - 客户端：`Windows7+` 或 `Linux 3.20+`（视发行版不同不同）
    - 服务端：`docker` 环境
    - 数据库：`mariadb` 或其他兼容的数据库系统。
2. 开发环境：
    - `windows` 和 `linux` 开发环境，缺乏 `mac` 设备，其他具体无限制
    - 软件环境无限制
    - 无专用硬件设备

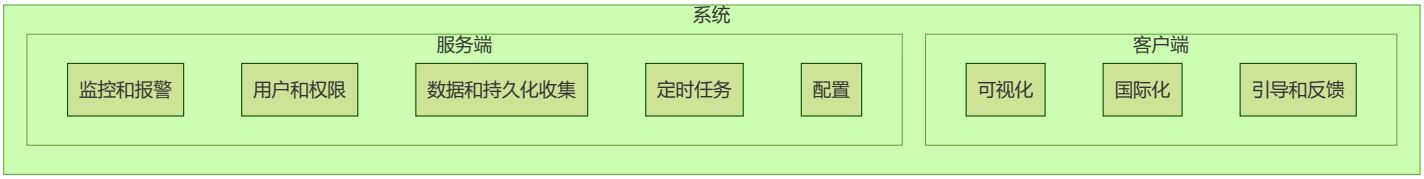
2.3.2 对外部的前提

1. GITHUB 用户系统基于 `OAuth 2.0` 协议，并提供了其 `事件流 API`，并且未在主站提供反爬虫。
2. 所有服务端设置在同一私有网络中。
3. 服务端运行与基于 `Linux` 的 `Docker` 虚拟化环境中。
4. 服务端有外网网关，固定 IP。

三. 功能需求描述

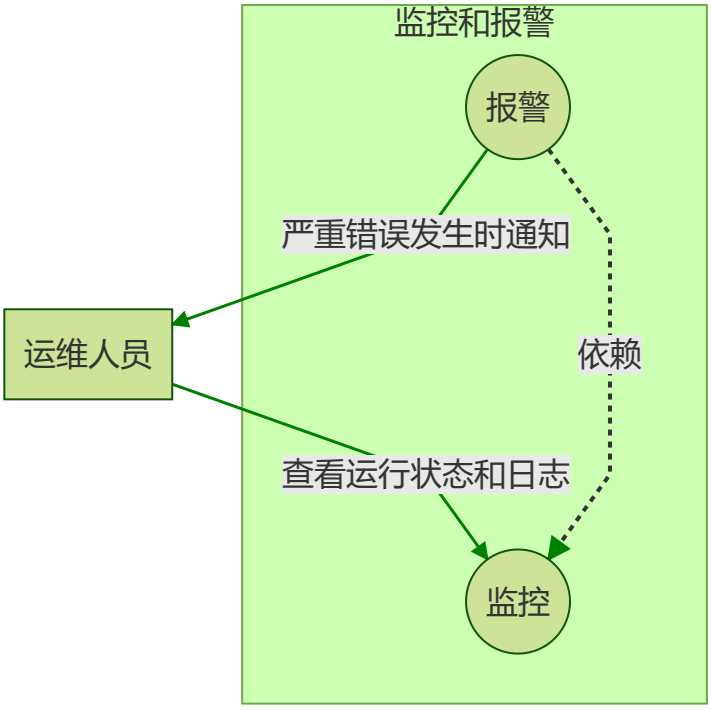
3.1 系统的划分

仅涉及实现不涉及需求的部分此处省略



3.2 监控和报警子系统的功能

3.2.1 用例图



3.2.2 用例

监控

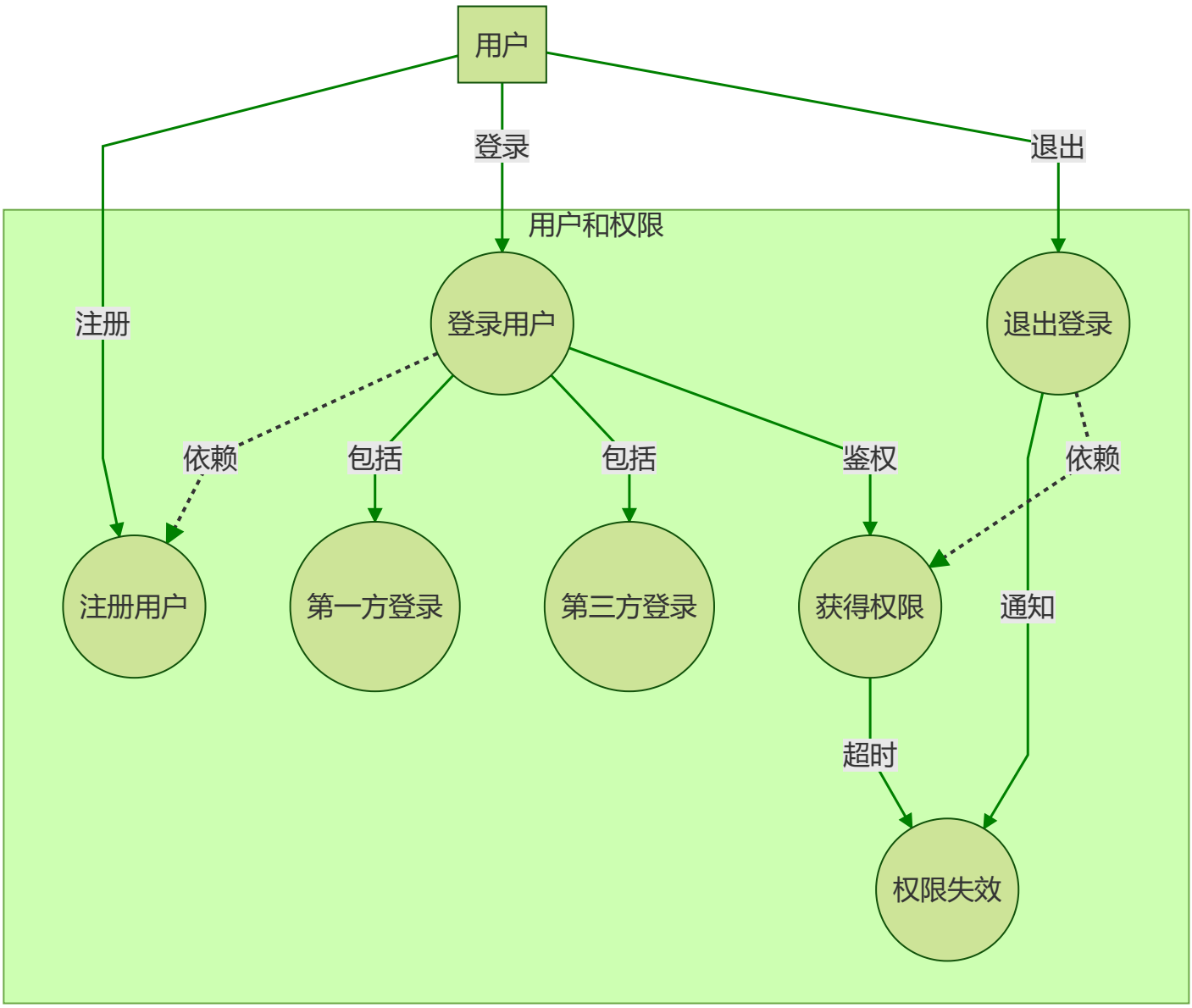
在系统正常运行情况下，运维人员可以通过监控模块查看系统运行情况和运行历史日志来判断和优化系统运行情况。

报警

在系统发生严重异常的情况下，监控模块会检测到异常的状态，并通过报警系统联系运维人员进行应急处理。

3.3 用户和权限系统的功能

3.3.1 用例图



3.3.2 用例

注册

软件用户在用户系统中不存在，可以注册，以使用软件。

登录

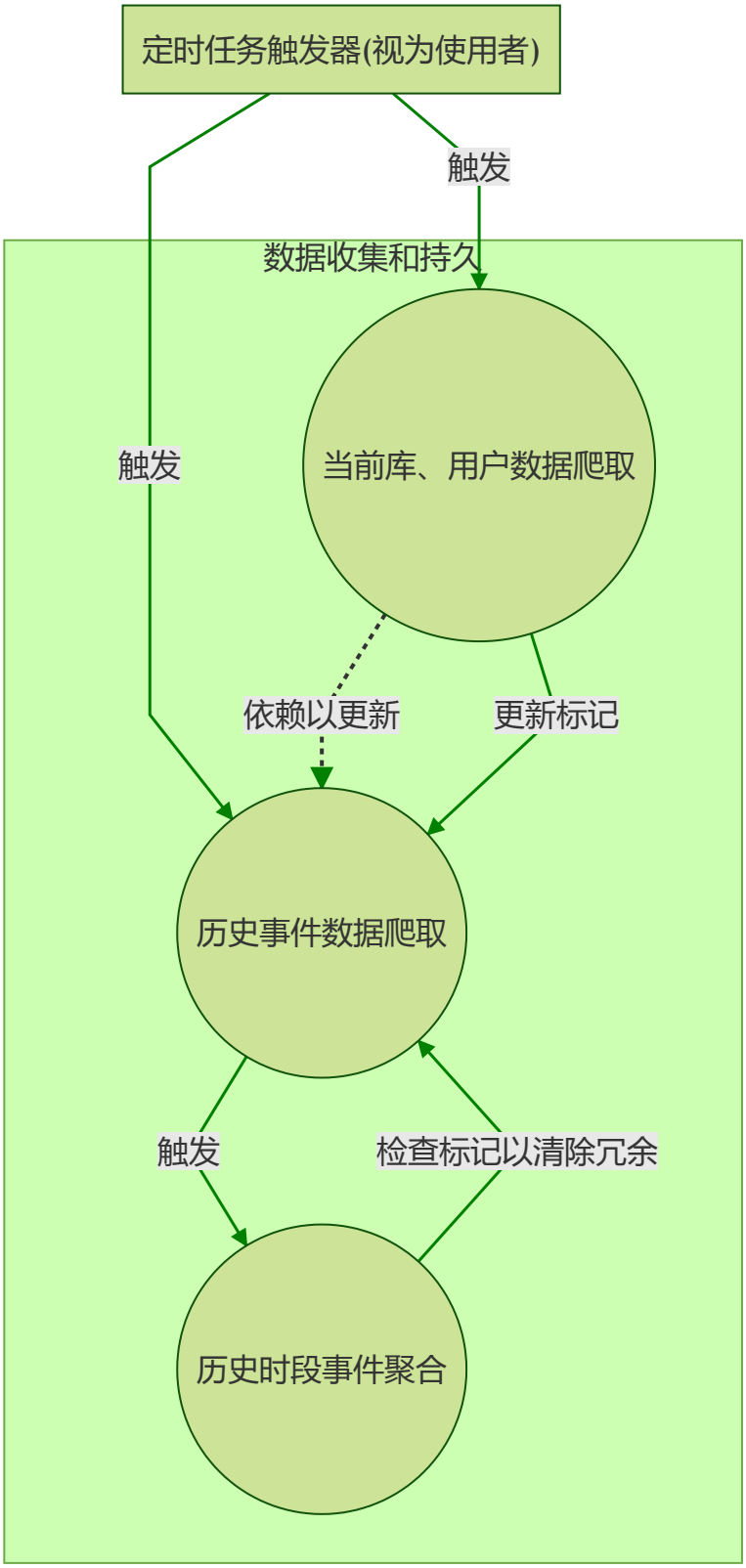
软件用户在用户系统中已经注册，通过第一方登录和第三方登录等方式可以登录，一旦鉴权成功，可以获得该用户的权限，以使用高级功能。

退出和超时

获得权限的软件用户，在权限超时和选择退出登录的情况下会权限失效，以保证安全性和正常离开系统。

3.3 数据收集和持久化子系统

3.3.1 用例图



3.3.2 用例

历史事件爬取

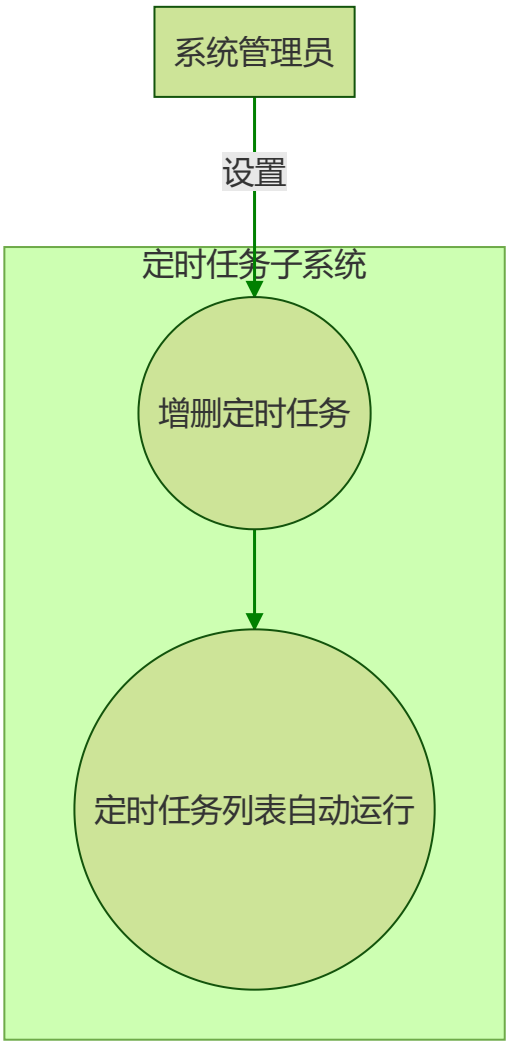
定时触发器触发历史事件爬取（增量事件爬取），当规定的时段爬取完毕后进行聚合，聚合完后检查上次聚合完毕后已做标记的原始数据进行删除，以腾出数据空间。

当前数据爬取

定时触发器触发当前数据爬取（全量数据爬取），检查历史事件爬取库中自上次爬取数据以来增量更新的对象，全量爬取其信息并更新，爬取后进行标记。

3.4 定时任务子系统功能

3.4.1 用例图

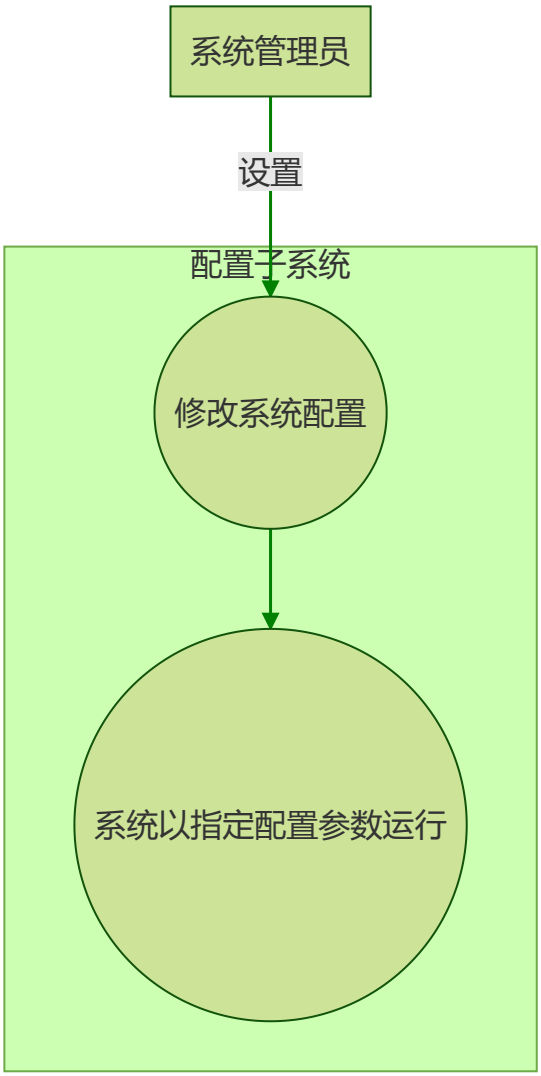


3.4.2 用例

系统管理员增删以编辑定时任务，定时任务列表中的任务会按照指定时间自动运行，自动化运行解放了管理员和运行维护人员的双手。

3.5 配置子系统

3.5.1 用例图

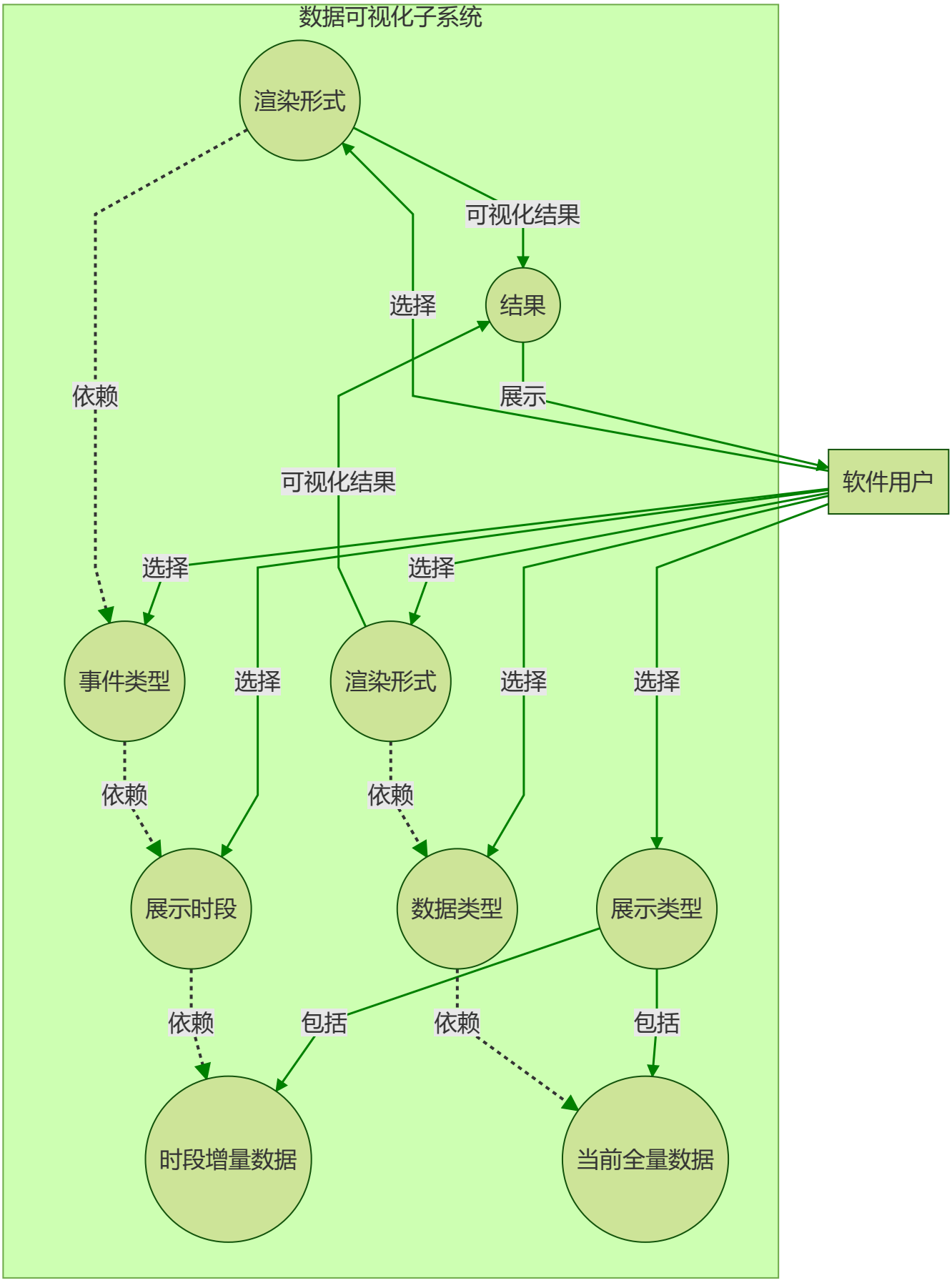


3.5.2 用例

系统管理员修改系统配置文件，系统会按照指定的配置运行，可配置系统增强了灵活性、适应性和可调节性，避免修改写死在代码中的数据可能带来的一系列问题。

3.6 数据可视化子系统

3.6.1 用例图



3.6.2 用例

时段增量数据可视化

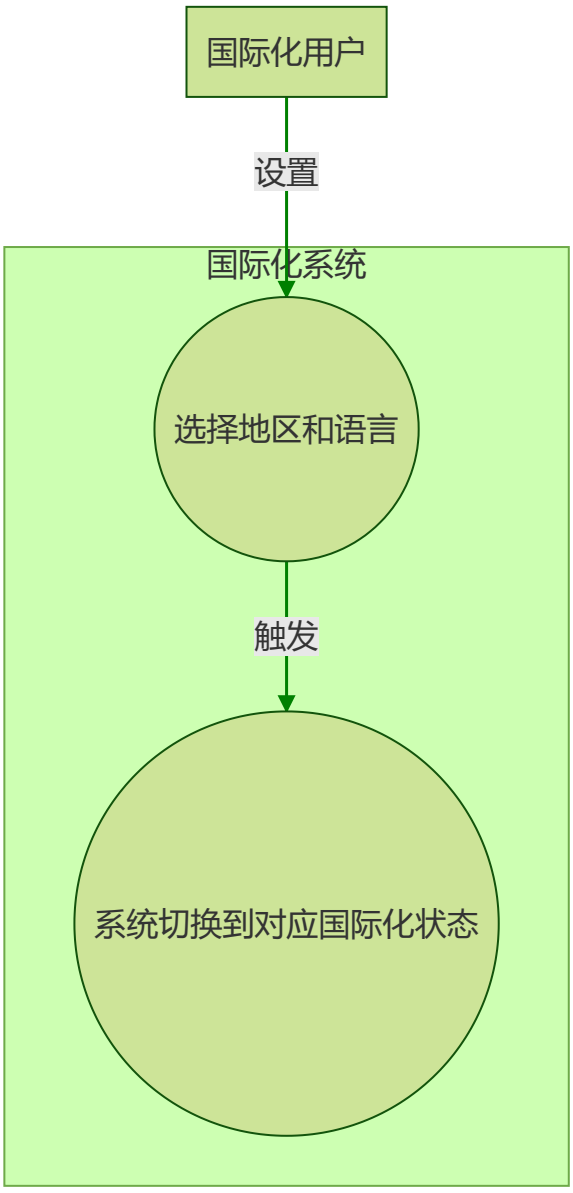
用户需要查看历史的增量数据，选择展示类型为时段增量数据可视化，选择展示时段，选择事件类型，选择渲染形式以获得可视化结果。

当前全量数据可视化

用户需要查看当前的全量数据，选择展示类型为当前全量数据可视化，选择数据类型，选择渲染形式以获得可视化结果。

3.7 国际化系统

3.7.1 用例图

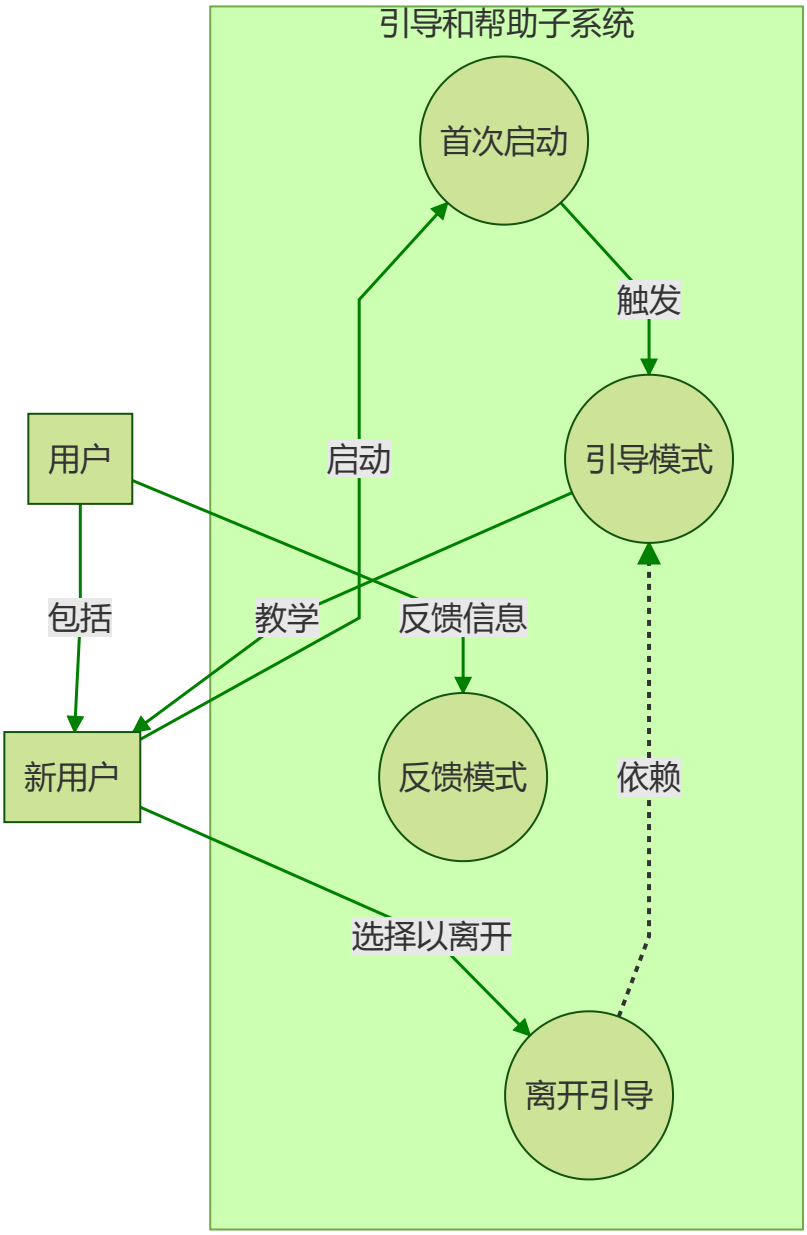


3.7.2 用例

国际化用户为了能够使用对应语言和地区设置的客户端软件，选择对应的地区和语言，触发系统切换到对应国际化状态。

3.8 引导和反馈子系统

3.8.1 用例图



3.7.2 用例

引导

软件的首次使用者，可以希望对软件的功能分区、使用方法可以有一个大体的了解，以便于对软件快速上手。如果已经通过其他渠道了解了使用方法，我也可以选择随时跳过这个引导，以节省时间。

反馈

对软件有任何意见和建议可以对开发者留言。

四. 其它需求描述

4.1 性能要求

4.1.1 响应性能

该软件为非实时性应用软件，无硬实时、软实时响应性能约束。只要软件能够在不影响用户体验的延迟时间内响应即可。

4.1.2 运行效率

该软件系统为服务端集中计算，客户端提供可视化和交互。

数据库

主数据库（主体数据存储）以PostgreSQL为实例，容量为 `100 GiB`，副数据库（用户和其它系统数据）容量为 `50 GiB`，处理机单核心 `2 GiB` 主存，在暂时无法分库分表情况下，能够在亿级大表上达到能够接受的查询时间

服务端

IO 密集型，要求有较高的并发 IO 能力和效率（单核万级）；计算较为密集，单次运行（每次更新）需要保证有接近 100% 的至少 1 核 CPU 计算资源，`1.5 GiB` 空闲内存。其它服务内存要求较低，核心服务内存要求在 `200 MiB` 左右，运行辅助服务在 `100 MiB - 200 MiB` 之间。

客户端

客户端性能要求较为普通，为当下市场中主流 Web 转制应用级别。要求在主流的民用级 PC 上以中低负载运行，并能够保持肉眼可见的流畅交互和舒适的日常使用。内存要求在 `N * 100MiB` 级别。

4.2 设计约束

4.2.1 开发工具

甲方对于具体开发工具无要求，但以市场主流工具栈为佳，便于后续维护和开发。

4.2.2 运行环境

服务端

运行在 `docker` 中，系统平台建议选用 `Linux`，便于支持前者的半虚拟化，否则需要启用虚拟化来模拟 `docker` 环境，造成较大性能损耗。至少有独占 1 核处理器，`2 GiB` 运行内存。

服务端网络需要有在广域网上的固定 IP，与用户规模匹配的上行带宽，与爬取数据规模匹配的下行带宽。

以上要求均为最低要求，可能随软件更新、用户规模变化、数据规模变化增长。

客户端

对于必要发行的 `linux` 和 `windows` 版本，需要有对应的系统环境。

其中前者需要图形支持和内核支持，由于 `linux` 发行版众多、依赖情况复杂、使用者多具备一定相关知识，支持情况具体参阅 `electron` 的 electron 官方文档。

对于 `windows`，至少需要 `windows7` 以上的系统版本，且官方发行的工具完备版本 —— 第三方编辑后的版本，由于可能缺失工具链无法运行。（ 例如：`ghost` 版本和 `非法 kms 激活` 版本，这些版本不安全、难于使用、不合法 ）

客户端将只发行 `X64` 架构的版本，用户主机需要满足市场主流设备性能和对应架构。

4.2.3 安全性

主要分为：

1. 通讯安全：保证数据和信息在传输过程中的基本安全。
2. 认证安全：保证用户系统在认证上的安全性。
3. 数据安全：保证密码等私密信息在服务端/数据库持久存储上的安全。

技术细节参见 2.2.10 安全和保密方面的考虑 章节。

4.2.4 可靠性

服务端

- 对于服务端短期的内存耗尽或者被系统杀死导致的应用崩溃 —— 一类的可恢复错误，可被监控系统在较短且适当的时间内恢复。
- 对于服务端遇到的未知类型严重错误导致的 `panic` 等严重错误，崩溃后由监视系统通知运维人员。

客户端

- 在服务端正常，用户正常使用情况下，不因自身原因影响可靠性，导致出现严重 `bug`。
- 在服务端因特殊情况而无法工作情况下，能够检测并主动告知用户，而非浪费用户宝贵的时间。

4.3 界面要求

4.3.1 界面设计导向

自然

1. 在行为的执行中，辅助用户有效决策，减少用户额外操作，让人机交互更加自然。
2. 在感知和认知中，尽量提取自然元素加强产品体验。

确定

1. 保持克制：不过多的堆砌无用元素，而是将焦点聚集在产品的核心价值，用精简凝练的设计元素加以表达。
2. 统一的系统化的设计：保持界面设计风格、设计、行为的一致性。

4.3.2 界面设计原则

关联度与距离

关联度高的信息总体更为接近，更应该形成一个视觉单元，从而实现高度组织性。

整齐

通过合适的对齐手段和图形设计，形成整齐的布局，引导用户视觉流向，让用户更加流畅的接收信息。

突出和对比

通过提高对比效应的原理，提升元素的组织性，让用户快速识别到关键信息。例如重要的行为流对应的按钮有突出的颜色。

统一和重复

统一的行为习惯、设计语言和重复的行为模式帮助用户熟悉使用、识别元素关联。

简洁和直截

使得操作直观、易于理解、易于使用，减少用户关注点的转移和重新聚焦引起的心智负担。

引导和过渡

通过一定程度的引导帮助用户熟悉软件，提供易发现性，并在通过过渡效果保持沟通性和生动性。

4.4 进度要求

在无突发事件的情况下，进度安排如下：

1. 第一轮迭代完成时，基础设施，应用框架，用户系统、监控系统已经完成（至少后端部分完成），附加的单元测试已经通过 —— 完成的子系统能独立完成应有功能（缺失内部依赖 mock 模拟）。
2. 第二轮迭代完成时，完成全部数据获取，能够自动化运行，进行细节功能的填充和各子系统的对接，形成可用的 demo。
3. 测试、集成及后处理阶段，排除前序阶段中没有排除的错误，集成整个系统，在保证完整性的基础上进行集成测试，并检查 checklist。

4.5 交付要求

编号	交付项	描述
1	程序	包括软件的安装程序与软件源代码
2	支持文件与环境说明、清单	第三方库、第三方插件以及必要的开发包和环境配置
3	文档	软件的说明文档，包括主要功能实现以及代码的说明(备注)等
4	报告	包括报告与演示PPT

4.6 验收要求

4.6.1 课件要求的基本项目功能点

基本需求指标

功能点	描述	备注
爬取数据	包括 Github 一定时间段内可爬取活动的数据	基本信息包括用户信息、 仓库信息等
账户登陆	实现账户的注册/登录	注册时应实现至少一种验证码
报表功能	实现美观，便于使用、理解的报表	
数据可视化	通过各种手段、 各种形式实现数据可视化	用户可选择展示的内容
项目报告与答辩	制作报告与项目的展示	

基本测试指标

1. 较高的单元测试比率
2. 基本的集成测试
3. 基本的代码审查
4. 基本的安全性

4.6.2 软件质量指标的高级度量

高级需求指标

- 海量数据的采集、持久化、预处理
- 多维度的数据的挖掘、分析，结果报表的生成
- 优秀的美术设计、用户交互和可视化
- 用户系统、控制中心与反馈系统

- 优雅、可拓展的系统服务框架
- 跨平台的应用实现
- 原生的国际化
- 更多实现的高级功能（如自动化部署运维等）

高级测试指标

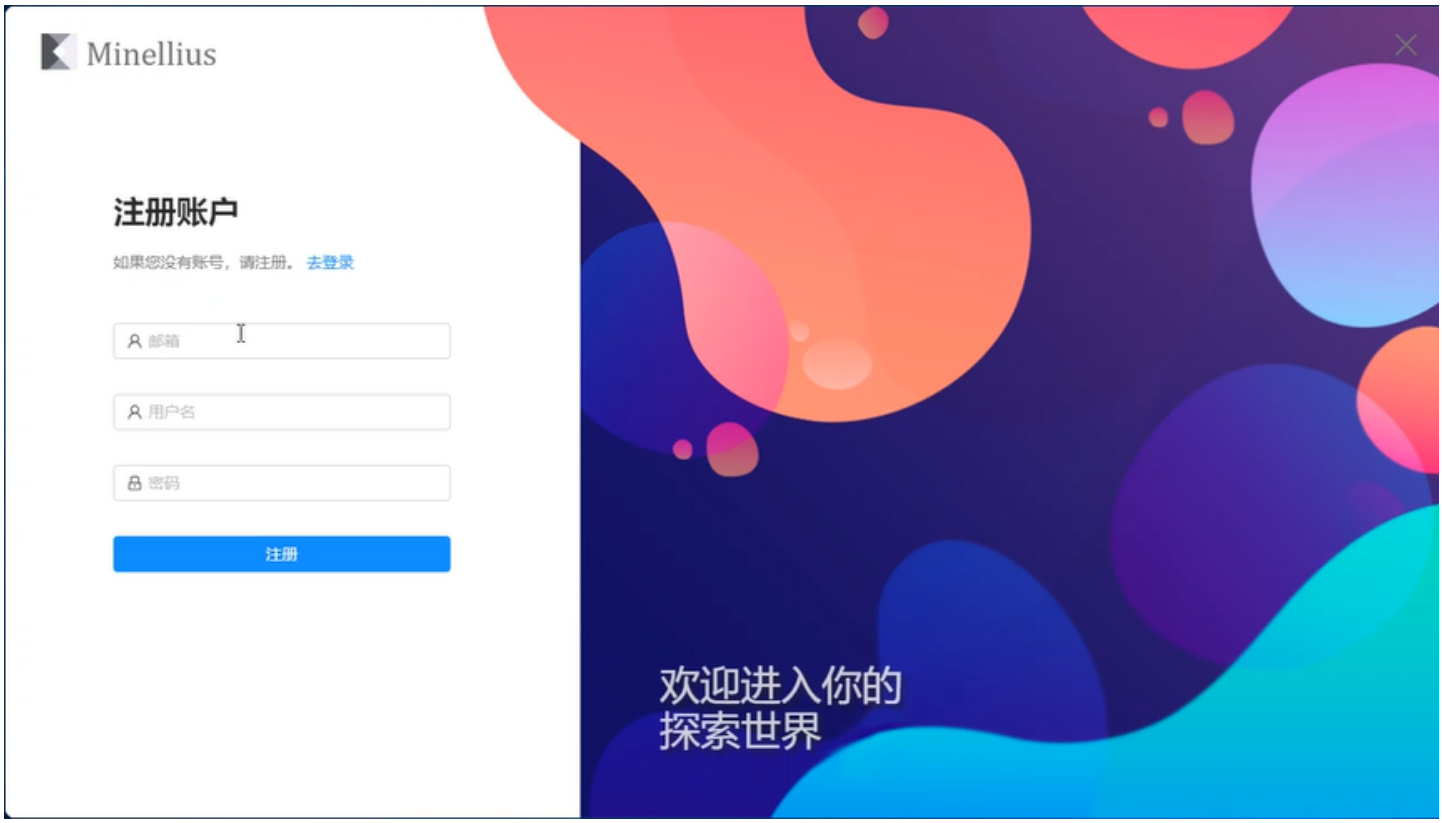
1. 完善的单元测试
2. 自动和人工结合的集成测试
3. 完善的代码审查
4. 较高的安全性（例如用户系统升级到jwt、TLS1.3等）
5. 较高的代码覆盖率

五. 软件原型

采集的数据示范

2	Azure-San	7
2	Talend	1
2	LaBetePolit	1
2	epicf	11
2	reserveMe	1
2	Pratilipi	227
2	linux-rdma	4
2	Infinideastu	1
2	knot-schol	1
2	mxe	1
2	VanClojure	4
2	MedicalAp	2
2	dovecot	1
2	elifescience	1
2	dOrgTech	7
2	codefornl	2
2	Railcraft	20
2	nu-book	2
2	nodejs	16
2	uport-proj	2
2	espressif	7
2	transloadit	3
2	FRC-Team	1

注册界面示范



使用交互示范


Minellius
首页概览
时段分析
总量分析
控制面板
帮助反馈
创库时间
库语言 B
Star数
Fork数
库体量
最后活跃
许可证
问题热度
所在地
所有库
关注度
库语言统计图
C
C#
C++
Haskell
HTML
css
GO
JAVA
JavaScript
MATLAB
Python
R
Ruby
Scala
PHP
Lua
Objective-C
Perl
Shell
Swift
Tex
Typescript
Rust
other
ActionScript
Clojure
CoffeeScript
1/2