吱星动力局

ZiZiPOWER

Rory Chou、叶子

Awesome-Bomb Studio

目录

CONTENTS

PART ONE

01

功能介绍

功能介绍---总览

PART ONE

用户管理体系

承担用户身份核验与个人信息全生命周期管理

宠物与设备关联系统

实现仓鼠档案与跑轮感应设备的绑定管理，支持多主体并行管理

聚焦仓鼠饲养管理与用户社交互动的专业化平台

首页数据展示与交互中心

整合核心监测功能与入口导航，分为四个功能区块

吱星社区生态平台

为用户提供内容创作与社交互动功能，由三个核心子系统构成

功能介绍---用户管理体系（1）

PART ONE

注册与认证

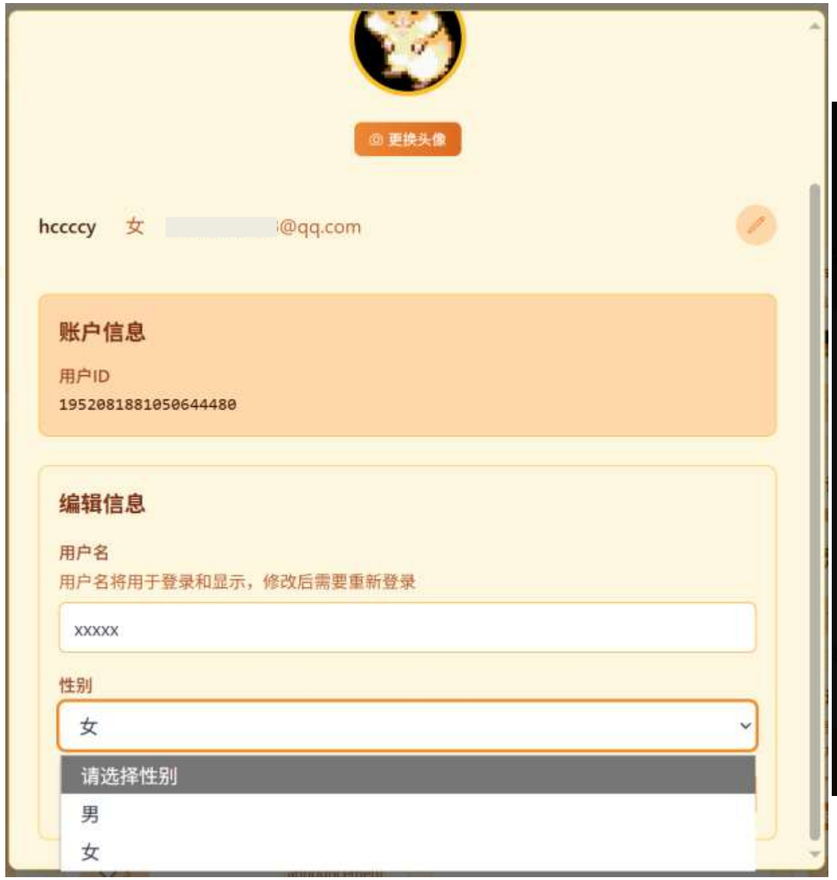
- 采用邮箱注册机制，系统通过发送验证码完成身份校验；
- 注册需提交邮箱、用户名（昵称）、密码及验证码（必填项），手机号为可选补充信息；
- 支持邮箱+密码以及邮箱+验证码两种登录模式

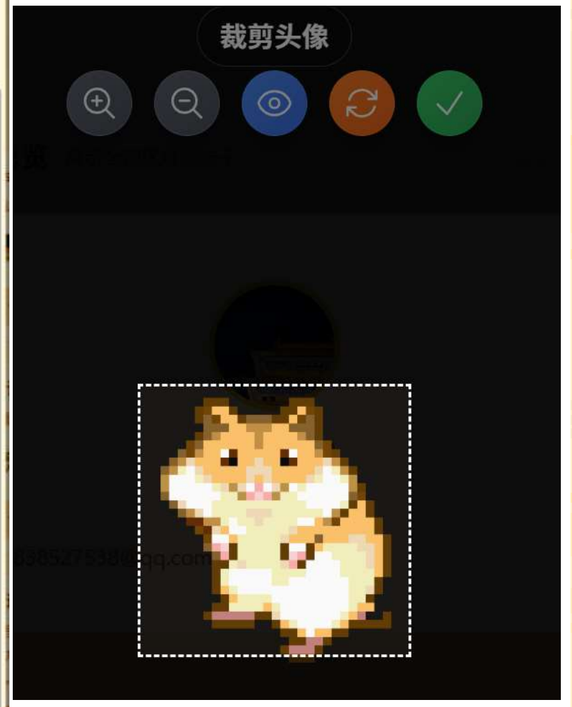
功能介绍---用户管理体系（2）

PART ONE

个人信息维护

- 支持已登录用户修改用户名等个人信息；
- 提供头像个性化设置功能，支持上传过程中的裁剪、缩放及实时预览





功能介绍---仓鼠与设备关联系统（1）

PART ONE

设备绑定机制

- 用户最多可关联 5 只仓鼠及对应的 5 个跑轮感应设备（设备号由官方分配，作为唯一绑定标识）；
- 数据采集方面，测试阶段通过模拟器生成运动数据（启动后每 20 秒推送一次，包含圈数、距离、速度及跑轮直径），正式应用环境对接物理传感器设备。

功能介绍---仓鼠与设备关联系统（2）

PART ONE

仓鼠档案管理

- 添加绑定仓鼠时需录入基础属性（姓名、性别、体重、出生日期等），支持宠物头像上传（含裁剪、缩放、预览功能）；
- 后期可对宠物属性及设备关联关系进行修改，确保信息时效性。

功能介绍---首页数据展示与交互中心（1）

PART ONE

实时运动监测面板

- 动态可视化组件实时反馈当前宠物的运动情况，动画速度与传感器（或模拟）数据联动；
- 面板底部同步展示关键指标：当前速度、累计圈数、今日最高速度，实现远程实时监控。

功能介绍---首页数据展示与交互中心（2）

PART ONE

运动排行榜系统

- 基于全平台宠物累计运动距离构建日榜、周榜和月榜三大榜单；
- 更新规则如下，日榜：统计当日累计距离，整点更新；周榜：统计当周累计距离，每日 0 点更新，支持查看本周及上周数据；月榜：统计当月累计距离，每日 0 点更新，支持查看本月及上月数据。

功能介绍---首页数据展示与交互中心（3\4）

PART ONE

仓鼠列表切换功能

展示用户已绑定的所有仓鼠档案摘要，支持用户通过列表选择目标仓鼠，切换至实时监测面板，实现多宠物状态的快速切换查看

吱星社区入口

提供“吱星社区”的快捷跳转通道

功能介绍---吱星社区生态平台（1）

PART ONE

内容分区导航

采用地图化分区展示帖子分类，共九个主题区域，点击可跳转至对应分类的帖子列表页，各区域名字和相关内容如下

吱星总部

展示全部区域的帖子

鼠宝教育

行为/互动/训练

撒欢基地

玩具/DIY/造景

鼠鼠秀场

萌图/摄影/颜值

健康中心

健康/病症/护理

鼠民街道

闲聊/树洞/碎碎念

吱吱广播

公告/规则/指南

鼠粮仓库

鼠粮/营养/食谱

跳蚤市场

交易/闲置/团购

功能介绍---吱星社区生态平台（2）

PART ONE

内容创作与互动管理

主要包含贴子发布功能和点赞互动功能

帖子发布功能

用户可选择目标分区，填写标题、正文，上传至多3张图片后发布，内容自动归入对应分区

帖子互动功能

- 展示对应分区的帖子，右下角标注所属分区；
- 支持浏览、点赞、查看详情及图片；
- 用户可删除自己发布的帖子，保障内容管理权限

功能介绍---吱星社区生态平台（3）

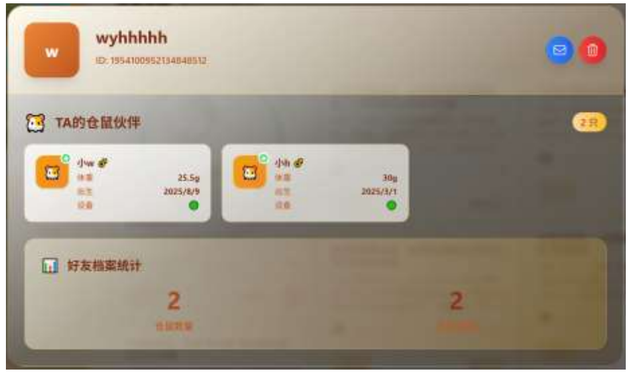
PART ONE

社交互动系统

主要包含好友管理和邮件通信功能

好友管理功能

- 通过用户 ID 或昵称检索并发送好友申请；
- 查看好友列表、待确认申请（发出 / 收到）；
- 点击好友卡片可查看其基础信息（昵称、ID）、关联宠物总览及档案统计（宠物数量、设备连接数）；
- 支持发起邮件或解除好友关系。



邮件通信功能

- 邮件分类查看（未读 / 已读 / 已发送）；
- 支持向好友列表用户撰写并发送邮件（需填写主题及正文）

PART TWO

02

创新点

创新点（1）---垂直服务深度

PART TWO

精准锁定垂直受众，构建仓鼠饲养者专属服务闭环

明确目标群体

聚焦“仓鼠饲养爱好者”这一细分群体，突破传统宠物类应用的泛化服务模式，围绕仓鼠饲养的核心需求（健康监测、互动社交、知识获取）打造全场景解决方案。

全链路服务闭环

通过整合设备绑定、运动监测、社区交流等功能，实现从“宠物管理”到“兴趣社交”的全链路服务，填补了仓鼠饲养领域专业化、集成化平台的空白。

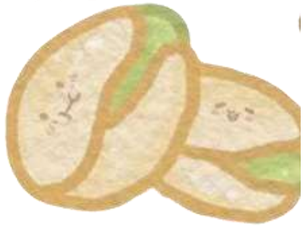
创新点（2）---功能设计融合性

PART TWO

硬件-数据-交互的创新融合，重构宠物监测新体验

实时监测的可视化创新

将跑轮感应设备采集的运动数据与可视化动画结合，通过“仓鼠跑轮运动”的具象化呈现，让抽象数据转化为直观可感知的场景，解决了远程宠物状态监测的“数据枯燥性”问题，提升用户实时互动感和应用的趣味性。



多主体管理的灵活性设计

支持最多5只仓鼠及对应设备的绑定管理，兼顾多宠物用户的实际需求；同时允许宠物档案及头像的个性化定制，强化用户对宠物的“专属感”，并且提高管理便捷性。

创新点（3）---用户体验场景化

PART TWO

公平竞技与社交生态的创意联动，增强用户粘性

科学的排行榜机制

采用“运动距离”而非“圈数”作为排行指标，规避跑轮直径差异导致的不公平，并设置日/周/月多维度榜单及历史查询功能，既保证竞技公平性，又通过阶段性目标激励用户持续关注宠物运动状态。

场景化社区生态

以“吱星地图”为核心的分区设计将论坛内容分区分类，既满足用户精准获取信息的需求，又降低用户探索门槛；同时，社交功能与宠物信息深度绑定，让社交围绕“仓鼠培养”这一共同兴趣展开，强化社区凝聚力。

PART THREE

03

技术架构

技术架构---前端技术栈(1)

PART THREE

React

React 是 Facebook 推出的声明式组件化 UI 库，核心思想为将 UI 拆分成可复用的组件，通过 Virtual DOM diff 算法高效更新视图。

src/main.jsx 中挂载根组件：

```
src > main.jsx
1  import { StrictMode } from 'react'
2  import { createRoot } from 'react-dom/client'
3  import './index.css'
4  import App from './App.jsx'
5
6  createRoot(document.getElementById('root')).render(
7    <StrictMode>
8      <App />
     </StrictMode>,
   )
```

使用函数组件 + Hooks（useState、useEffect、useContext)：

```
const AuthProvider = ({ children }) => {
  const [user, setUser] = useState(null);
  const [loading, setLoading] = useState(true);
  const [isLoggedIn, setIsLoggedIn] = useState(false);
  const [userHamsters, setUserHamsters] = useState([]);
```

```
const context = useContext(AuthContext);
```

```
// 初始化认证状态
useEffect(() => {
```

技术架构---前端技术栈(2\3)

PART THREE

Vite

Vite 是基于原生 ES 模块的前端构建工具，开发模式下利用浏览器原生 ESM 加速冷启动；生产模式下基于 Rollup 做打包优化。

根目录 vite.config.js配置 React 插件、别名等：

```
vite.config.js > ...
1   import { defineConfig } from 'vite'
2   import react from '@vitejs/plugin-react'
```

package.json 脚本用 vite 启动开发服务器，vite build 产出静态文件：

```
"scripts": {
  "dev": "vite",
  "build": "vite build",
```

Tailwind CSS

Tailwind 是原子化 CSS 框架，通过类名直接控制样式，避免全局样式冲突，提升开发效率与可维护性。

tailwind.config.js 定制主题：

```
JS tailwind.config.js
JS tailwind.config.js > default > daisyui > themes > warm > "success"
1   /** @type {import('tailwindcss').Config} */
2   export default {
3     content: [
4       "./index.html",
5       "./src/**/*.{js,ts,jsx,tsx}",
6     ],
```

技术架构---前端技术栈(4\5)

PART THREE

前端路由（React-router)

SPA 单页应用通过客户端路由实现多页面切换，常用库 react-router 提供 <Routes>、<Route> 定义和 useNavigate 编程式导航。

App.jsx 中配置路由：

```jsx
return (
  <div className="App flex flex-col min-h-screen">
    {/* 只在用户已登录时显示导航栏 */}
    {isLoggedIn && <Navbar />}
    <main className="flex-grow">
      <Routes>
        <Route path="/" element={<HomePage />} />
        <Route
          path="/dashboard"
          element={
            <ProtectedRoute>
              <DashboardPage />
            </ProtectedRoute>
          }
        />
        <Route
          path="/zhixing"
          element={
            <ProtectedRoute>
              <ZhixingPage />
            </ProtectedRoute>
          }
        />
        <Route
          path="/admin"
          element={
            <AdminRoute>
              <AdminPage />
            </AdminRoute>
          }
        />
        <Route path="*" element={<Navigate to="/" replace />} />
      </Routes>
    </main>
  </div>
);
};
```

接口调用

HTTP 客户端库用于前后端数据交互，Axios 支持拦截器、自动 JSON 序列化等特性。

src/services/api.js 封装 axios 实例：

```jsx
// 创建axios实例
const api = axios.create({
  baseURL: API_BASE_URL,
  timeout: 10000,
  headers: {
    'Content-Type': 'application/json'
  }
});
```

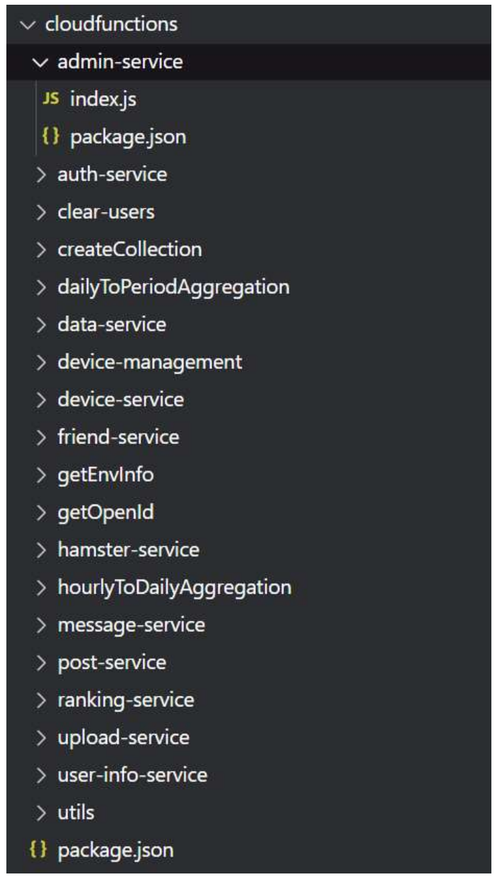
技术架构---后端技术栈(1\2)

PART THREE

腾讯云CloudBase（云函数）

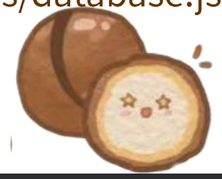
CloudBase 是腾讯云提供的无服务器（Serverless）云开发平台，通过云函数、云数据库、文件存储等服务实现后端功能，可按需弹性伸缩、免运维。

通过在cloudfunctions 目录下创建多个服务，云函数通过cloudbaserc.json 配置部署信息，并使用 tcb deploy 一键发布。



云数据库（TCB 直连MongoDB）

CloudBase 可接入多种数据库，一般使用其托管的 MongoDB （类 NoSQL）或融入关系型数据库。集合（collection）+ 文档（document）方式存储，灵活且易扩展。通过各服务 utils/database.js 中封装 db 操作实现。



database.js

cloudfunctions > utils > database.js > ...

```
1  const cloud = require('@cloudbase/node-sdk');
2
3  function getDatabase() {
4    return cloud.database();
5  }
6
7  module.exports = {
8    getDatabase
9  };
```

技术架构---后端技术栈(3\4)

PART THREE

认证与授权

常见 JWT（JSON Web Token）在前后端分离模式下管理会话，前端存储 token，后端验证签名确保安全。

auth-service 中生成并校验 JWT；各业务云函数 utils/auth.js 提供 verifyToken 中间件，于入口函数中调用，拒绝未授权请求。

```
class AuthService {
  /**
   * 生成JWT令牌
   */
  static generateToken(payload) {
    try {
      return jwt.sign(payload, JWT_SECRET, { expiresIn: '7d' });
    } catch (error) {
      console.error('生成JWT令牌失败:', error);
      throw error;
    }
  }
```

```
/**
 * 验证JWT令牌
 */
static verifyToken(token) {
  try {
    return jwt.verify(token, JWT_SECRET);
  } catch (error) {
    console.error('验证JWT令牌失败:', error);
    throw error;
  }
}
```

日志与监控

无服务器架构需依赖云平台日志系统记录函数调用、错误及性能指标。

云函数内使用 console.log、console.error 记录，Platform 控制台可查看执行日志。

技术架构---综合流程

PART THREE

2.客户端路由与导航

3.接口调用

4.云函数执行与数据库交互

1.资源加载与初始化

5.前端接收响应并更新视图

PART FOUR

04

心得体会

工具使用心得

PART FOUR

智能协同：从单兵作战到AI搭档

CodeBuddy实现了需求拆解秒级响应，将自然语言需求迅速拆解，直接落地为可执行代码框架。同时可实现技术栈无缝衔接，其可基于腾讯云生态，自动关联云函数、数据库连接等，避免在技术文档中反复检索，真正实现全栈需求，一键协同。

效率突破：压缩开发周期的时间魔法

作为个人开发者，CodeBuddy让我们在功能迭代和问题调试环节效率大幅提升，针对重复功能代码，CodeBuddy可基于过往代码模式生成模板，只需补充业务参数即可复用。同时针对某些问题可以进行智能诊断，避免了盲目试错。

成长加速：技术视野和开发思维的双重拓展

CodeBuddy 不止是“代码生成器”，更是“全栈能力放大器”，其帮助开发者突破技术边界，拓展了全栈开发的技术深度。同时促进其思维升级，让我们真正理解全栈开发中“技术与体验协同”的思维逻辑。

总结

CodeBuddy 不是替代开发者，而是让开发者从写代码的人，变成指挥代码的人，在全栈开发的战场上，真正实现“创意无上限，开发高效率”。

Thank You!

感谢观看

ZiZiPOWER