猪猪银行开发详细步骤指南 - M1 Mac版

第一部分：环境准备（约30分钟）

步骤1：安装Homebrew（如果还没安装）

打开终端（Terminal），输入以下命令：

```bash
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

步骤2：安装Node.js

```bash
# 使用Homebrew安装Node.js
brew install node

# 验证安装
node --version  # 应该显示 v18.x.x 或更高
npm --version   # 应该显示 9.x.x 或更高
```

步骤3：安装Visual Studio Code（推荐的代码编辑器）

```bash
# 使用Homebrew安装
brew install --cask visual-studio-code

# 或者从官网下载：https://code.visualstudio.com/
```

步骤4：创建项目文件夹

```bash
# 在桌面创建项目文件夹
cd ~/Desktop
mkdir piggy-bank-project
cd piggy-bank-project
```

第二部分：创建React应用（约15分钟）

步骤5：创建React项目

bash

```
# 使用create-react-app创建TypeScript项目
npx create-react-app piggy-bank --template typescript

# 进入项目文件夹
cd piggy-bank

# 在VS Code中打开项目
code .
```

步骤6：安装必要的依赖包

bash

```
# 安装UI组件库
npm install @mantine/core @mantine/hooks @mantine/notifications @mantine/dates

# 安装样式工具
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p

# 安装路由
npm install react-router-dom @types/react-router-dom

# 安装状态管理
npm install zustand

# 安装动画库
npm install framer-motion

# 安装图表库
npm install react-chartjs-2 chart.js

# 安装PWA支持
npm install workbox-webpack-plugin workbox-window
```

第三部分：配置项目基础（约20分钟）

步骤7：配置Tailwind CSS

打开 `tailwind.config.js` 文件，替换内容：

```javascript
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: [
    "./src/**/*.{js,jsx,ts,tsx}",
  ],
  theme: {
    extend: {
      colors: {
        'piggy-pink': '#FF69B4',
        'piggy-blue': '#4A90E2',
        'piggy-green': '#7ED321',
        'piggy-orange': '#F5A623',
        'piggy-cream': '#FFF8DC',
      }
    },
  },
  plugins: [],
}
```

步骤8：更新CSS文件

打开 `src/index.css`，在顶部添加：

```css
@tailwind base;
@tailwind components;
@tailwind utilities;
```

步骤9：创建项目文件结构

在终端中运行：

```bash
# 创建必要的文件夹
mkdir -p src/components
mkdir -p src/pages
mkdir -p src/services
mkdir -p src/stores
mkdir -p src/utils
mkdir -p src/types
mkdir -p public/icons
mkdir -p public/sounds
```

第四部分：创建PWA配置文件（约15分钟）

步骤10：创建Web App Manifest

创建文件 `public/manifest.json`：

```json
{
  "name": "猪猪银行",
  "short_name": "猪猪银行",
  "description": "儿童评分奖励系统",
  "start_url": "/",
  "display": "standalone",
  "background_color": "#FFF8DC",
  "theme_color": "#FF69B4",
  "orientation": "portrait",
  "icons": [
    {
      "src": "/icons/icon-192x192.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "/icons/icon-512x512.png",
      "sizes": "512x512",
      "type": "image/png"
    }
  ]
}
```

步骤11：创建Service Worker

创建文件 `public/service-worker.js`：

javascript

```javascript
/* eslint-disable no-restricted-globals */
const CACHE_NAME = 'piggy-bank-v1';
const urlsToCache = [
  '/',
  '/static/css/main.css',
  '/static/js/bundle.js',
];

self.addEventListener('install', event => {
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then(cache => cache.addAll(urlsToCache))
  );
});

self.addEventListener('fetch', event => {
  event.respondWith(
    caches.match(event.request)
      .then(response => {
        if (response) {
          return response;
        }
        return fetch(event.request);
      }
    )
  );
});
```

步骤12：更新HTML文件

打开 `public/index.html`，在 `<head>` 标签内添加：

html

```html
<link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
<meta name="theme-color" content="#FF69B4" />
<meta name="apple-mobile-web-app-capable" content="yes" />
<meta name="apple-mobile-web-app-status-bar-style" content="default" />
<meta name="apple-mobile-web-app-title" content="猪猪银行" />
```

第五部分：创建基础组件（约30分钟）

步骤13：创建类型定义

创建文件 `src/types/index.ts`：

typescript

```typescript
export interface ScoreCategory {
  id: string;
  name: string;
  icon: string;
  maxScore: number;
}

export interface DailyScore {
  id: string;
  date: string;
  category: string;
  score: number;
  notes?: string;
}

export interface Reward {
  id: string;
  name: string;
  description: string;
  cost: number;
  type: 'physical' | 'privilege' | 'virtual';
  image?: string;
}

export interface Achievement {
  id: string;
  name: string;
  description: string;
  icon: string;
  unlocked: boolean;
  unlockedDate?: string;
}
```

步骤14：创建状态管理

创建文件 `src/stores/useStore.ts`：

typescript

```typescript
import { create } from 'zustand';
import { persist } from 'zustand/middleware';

interface AppState {
  totalStars: number;
  dailyScores: any[];
  achievements: any[];
  addStars: (amount: number) => void;
  addDailyScore: (score: any) => void;
}

export const useStore = create<AppState>()(
  persist(
    (set) => ({
      totalStars: 0,
      dailyScores: [],
      achievements: [],
      addStars: (amount) => set((state) => ({
        totalStars: state.totalStars + amount
      })),
      addDailyScore: (score) => set((state) => ({
        dailyScores: [...state.dailyScores, score]
      })),
    }),
    {
      name: 'piggy-bank-storage',
    }
  )
);
```

步骤15：创建主页组件

创建文件 `src/pages/HomePage.tsx`：

typescript

```tsx
import React from 'react';
import { useStore } from '../stores/useStore';

export const HomePage: React.FC = () => {
  const { totalStars } = useStore();

  return (
    <div className="min-h-screen bg-piggy-cream p-4">
      <div className="max-w-4xl mx-auto">
        <header className="text-center py-8">
          <h1 className="text-4xl font-bold text-piggy-pink mb-4">
            🐷 猪猪银行 🐷
          </h1>
          <div className="bg-yellow-400 text-white px-6 py-3 rounded-full inline-block">
            <span className="text-2xl font-bold">⭐ {totalStars} 星星</span>
          </div>
        </header>

        <div className="grid grid-cols-1 md:grid-cols-2 gap-6 mt-8">
          <div className="bg-white rounded-2xl shadow-lg p-6">
            <h2 className="text-2xl font-bold text-piggy-blue mb-4">
              今日任务
            </h2>
            <div className="space-y-3">
              <div className="flex items-center justify-between p-3 bg-piggy-cream rounded-lg">
                <span>📚 完成数学作业</span>
                <button className="bg-piggy-green text-white px-4 py-2 rounded-lg">
                  完成
                </button>
              </div>
              <div className="flex items-center justify-between p-3 bg-piggy-cream rounded-lg">
                <span>🏃 运动30分钟</span>
                <button className="bg-piggy-green text-white px-4 py-2 rounded-lg">
                  完成
                </button>
              </div>
            </div>
          </div>

          <div className="bg-white rounded-2xl shadow-lg p-6">
            <h2 className="text-2xl font-bold text-piggy-blue mb-4">
              本周进度
            </h2>
            <div className="mb-4">
              <div className="flex justify-between text-sm mb-2">
                <span>完成度</span>
```

```
            <span>80%</span>
          </div>
          <div className="w-full bg-gray-200 rounded-full h-4">
            <div
              className="bg-piggy-green h-4 rounded-full transition-all duration-500"
              style={{ width: '80%' }}
            />
          </div>
        </div>
        <p className="text-gray-600">
          再完成2个任务就能获得周奖励!
        </p>
      </div>
    </div>
  </div>
</div>
  );
};
```

步骤16：更新App.tsx

替换 `src/App.tsx` 的内容：

```typescript
import React from 'react';
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import { HomePage } from './pages/HomePage';

function App() {
  return (
    <Router>
      <Routes>
        <Route path="/" element={<HomePage />} />
      </Routes>
    </Router>
  );
}

export default App;
```

第六部分：运行和测试（约10分钟）

步骤17：启动开发服务器

```bash
# 在项目根目录运行
npm start

# 浏览器会自动打开 http://localhost:3000
```

步骤18：测试PWA功能

1. 打开Chrome浏览器
2. 访问 http://localhost:3000
3. 打开开发者工具（Command + Option + I）
4. 点击"Application"标签
5. 查看"Manifest"和"Service Workers"部分

第七部分：创建应用图标（约20分钟）

步骤19：创建应用图标

1. 使用任何图片编辑工具（如Preview、Figma等）创建一个猪猪图标
2. 保存为PNG格式，尺寸为512x512像素
3. 使用在线工具生成不同尺寸：https://www.pwabuilder.com/imageGenerator

步骤20：添加图标文件

将生成的图标文件放入 `public/icons/` 文件夹：

- icon-192x192.png
- icon-512x512.png
- apple-touch-icon.png

第八部分：构建生产版本（约10分钟）

步骤21：构建项目

```bash
# 构建生产版本
npm run build

# 安装本地服务器测试生产版本
npm install -g serve

# 运行生产版本
serve -s build

# 访问 http://localhost:3000 测试
```

第九部分：部署到网络（约20分钟）

步骤22：使用Vercel部署（免费方案）

```bash
# 安装Vercel CLI
npm install -g vercel

# 登录Vercel（会在浏览器中打开）
vercel login

# 部署项目
vercel --prod

# 按照提示操作，会得到一个URL如：
# https://piggy-bank-xxxxx.vercel.app
```

第十部分：生成手机应用（约30分钟）

步骤23：生成Android应用

```bash
# 安装PWA转应用工具
npm install -g @bubblewrap/cli

# 初始化（使用你的Vercel URL）
bubblewrap init --manifest=https://your-app.vercel.app/manifest.json

# 填写信息：
# - 应用名称：猪猪银行
# - 包名：com.yourname.piggybank
# - 其他保持默认

# 构建APK
bubblewrap build

# 生成的APK文件在当前目录
```

步骤24：生成iOS应用

1. 访问 https://www.pwabuilder.com/
2. 输入你的Vercel URL
3. 点击"Start"
4. 选择"iOS"平台
5. 下载生成的项目文件
6. 需要Xcode来编译（需要Apple Developer账号）

步骤25：生成Mac桌面应用

```bash
# 在项目目录安装Electron
npm install --save-dev electron electron-builder

# 创建electron主文件
touch public/electron.js
```

在 `public/electron.js` 中添加：

javascript

```javascript
const { app, BrowserWindow } = require('electron');
const path = require('path');

function createWindow() {
  const win = new BrowserWindow({
    width: 1200,
    height: 800,
    webPreferences: {
      nodeIntegration: false,
      contextIsolation: true
    },
    icon: path.join(__dirname, 'icons/icon-512x512.png')
  });

  win.loadURL(
    app.isPackaged
      ? `file://${path.join(__dirname, '../build/index.html')}`
      : 'http://localhost:3000'
  );
}

app.whenReady().then(createWindow);
```

在 package.json 中添加：

json

```json
"main": "public/electron.js",
"scripts": {
  "electron": "electron .",
  "dist": "npm run build && electron-builder"
},
"build": {
  "appId": "com.yourname.piggybank",
  "productName": "猪猪银行",
  "directories": {
    "output": "dist"
  },
  "mac": {
    "category": "public.app-category.education"
  }
}
```

构建Mac应用：

```bash
npm run dist
# Mac应用会在 dist/ 文件夹中
```

常见问题解决

问题1：npm命令找不到

```bash
# 重新加载终端配置
source ~/.zshrc
```

问题2：权限错误

```bash
# 使用sudo（需要输入密码）
sudo npm install -g [包名]
```

问题3：端口被占用

```bash
# 修改端口
PORT=3001 npm start
```

问题4：M1芯片兼容性问题

```bash
# 使用Rosetta模式运行
arch -x86_64 npm install [包名]
```

下一步建议

1. **添加更多功能**
   - 实现评分系统
   - 添加奖励兑换
   - 创建家长管理界面
2. **美化界面**
   - 添加动画效果
   - 设计可爱的卡通形象

- 优化颜色搭配

3. **数据存储**
   - 集成Firebase
   - 实现云端同步
   - 添加用户登录
4. **发布应用**
   - 申请开发者账号
   - 提交应用商店审核
   - 推广应用

需要帮助时，可以随时询问具体步骤的详细说明！