# HarmonyOS应用开发综合设计报告

# 一、前言

## 1. 项目背景及意义

随着万物互联时代的到来，HarmonyOS作为华为自主研发的分布式操作系统，为各类智能设备提供了统一的操作系统平台。本项目基于HarmonyOS应用开发框架，采用ArkTS语言进行应用开发，探索在分布式架构下的应用开发模式与实践。

## 相关技术说明

HarmonyOS: 华为自主研发的分布式操作系统

ArkTS: 基于TypeScript的声明式开发语言

Hvigor: HarmonyOS项目构建工具

OHPM: HarmonyOS包管理工具

ETS: 基于TypeScript的扩展开发语言

# 二、概要设计

## 1. 需求描述

本项目旨在开发一个符合HarmonyOS应用开发规范的基础应用，具备以下核心需求：

①首页新闻列表：显示新闻标题、摘要、来源、发布时间、缩略图；支持下拉刷新、上拉加载更多

②新闻分类：提供分类标签（如首页、国内、国际、科技）；点击切换分类

③新闻详情页：完整展示新闻标题、正文、图片、来源、发布时间，排版清晰易读

④收藏功能：在详情页可收藏/取消收藏，收藏状态有视觉反馈

⑤我的收藏页：展示所有收藏新闻，支持点击阅读与取消收藏

## 2. 整体开发设计架构

应用入口：EntryAbility.ts

View：对应pages下的各个页面组件

ViewModel：我为每个页面创建对应的ViewModel，用于处理页面逻辑

Model：对应model目录下的数据模型

Service：负责网络请求和本地数据存储

模块配置文件：module.json5

# 三、详细设计

## 1. 功能实现分析

通过.json5文件配置应用的基本信息、构建模式和模块依赖：

1. 模块定义配置：

{

"module": {

"name": "entry",

"type": "entry",

"description": "新闻阅读应用主模块",

"mainElement": "EntryAbility",

"deviceTypes": ["phone", "tablet"],

"deliveryWithInstall": true,

"installationFree": false,

"pages": "$profile:main\_pages",

}}

（2） 权限安全管理：

"requestPermissions": [

{

"name": "ohos.permission.READ\_IMAGEVIDEO",

"reason": "用于加载和显示新闻中的图片内容",

"usedScene": {

"abilities": ["EntryAbility"],

"when": "inuse"

}

},

{

"name": "ohos.permission.WRITE\_IMAGEVIDEO",

"reason": "用于缓存新闻图片和用户头像保存",

"usedScene": {

"abilities": ["EntryAbility"],

"when": "inuse"

}

}]

## 调用关系说明

构建流程：

Hvigor构建工具 → 读取build-profile.json5 → 配置模块和产品 → 执行构建任务

代码检查：

开发阶段 → ESLint规则检查 → 安全规则验证 → 输出检查结果

依赖解析：

OHPM包管理器 → 读取oh-package.json5 → 下载依赖包 → 生成锁文件

# 四、软件测试

## 1. 测试用例设计

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1.1 新闻浏览功能  | 用例ID | 测试场景 | 前置条件 | 测试步骤 | 预期结果 | 优先级 | | --- | --- | --- | --- | --- | --- | | TC-FN-001 | 首页新闻列表加载 | 网络正常 | 1. 启动App 2. 进入首页 | 1. 显示新闻列表 2. 包含标题、摘要、来源、时间、图片 | 高 | | TC-FN-002 | 新闻分类切换 | 首页已加载 | 1. 点击分类标签 2. 选择"科技"分类 | 1. 列表内容切换为科技新闻 2. 分类标签状态更新 | 高 | | TC-FN-003 | 下拉刷新 | 首页有内容 | 1. 在列表顶部下拉 | 1. 显示刷新动画 2. 列表内容更新 | 中 | | TC-FN-004 | 上拉加载更多 | 列表有分页 | 1. 滚动到底部 2. 继续上拉 | 1. 加载更多数据 2. 列表长度增加 | 中 |  1.2 新闻详情功能  | 用例ID | 测试场景 | 前置条件 | 测试步骤 | 预期结果 | 优先级 | | --- | --- | --- | --- | --- | --- | | TC-FN-005 | 打开新闻详情 | 首页有新闻 | 1. 点击任意新闻项 | 1. 跳转到详情页 2. 显示完整内容 3. 图片正常加载 | 高 | | TC-FN-006 | 详情页收藏功能 | 详情页打开 | 1. 点击收藏按钮 2. 再次点击 | 1. 按钮状态变化 2. 收藏数据保存 | 高 |  1.3 收藏功能  | 用例ID | 测试场景 | 前置条件 | 测试步骤 | 预期结果 | 优先级 | | --- | --- | --- | --- | --- | --- | | TC-FN-007 | 查看我的收藏 | 有收藏记录 | 1. 进入"我的"页面 2. 点击"我的收藏" | 1. 显示收藏列表 2. 可点击进入详情 | 高 | | TC-FN-008 | 取消收藏 | 收藏页有数据 | 1. 在收藏页取消收藏 2. 返回详情页检查 | 1. 列表项移除 2. 详情页收藏状态同步 | 中 |  1.4 用户功能  | 用例ID | 测试场景 | 前置条件 | 测试步骤 | 预期结果 | 优先级 | | --- | --- | --- | --- | --- | --- | | TC-FN-009 | 用户登录 | 未登录状态 | 1. 点击登录 2. 输入账号密码 3. 提交 | 1. 登录成功 2. 跳转到首页 3. 用户状态更新 | 高 | | TC-FN-010 | 用户注册 | 新用户 | 1. 进入注册页 2. 填写信息 3. 提交 | 1. 注册成功 2. 自动登录 3. 跳转首页 | 高 | | TC-FN-011 | 修改密码 | 已登录 | 1. 进入修改密码页 2. 输入原密码和新密码 3. 提交 | 1. 修改成功提示 2. 需要重新登录 | 中 | |

## 测试结果分析

| 测试用例ID | 测试场景 | 测试结果 | 发现 |
| --- | --- | --- | --- |
| TC-FN-001 | 首页新闻列表加载 | 通过 | 所有信息完整显示，首次点击加载慢 |
| TC-FN-002 | 新闻分类切换 | 通过 | 分类切换流畅，内容匹配准确 |
| TC-FN-003 | 下拉刷新 | 通过 | 刷新机制成功运行 |
| TC-FN-004 | 上拉加载更多 | 通过 | 刷新机制成功运行 |
| TC-FN-005 | 打开新闻详情 | 通过 | 页面跳转流畅 |
| TC-FN-006 | 详情页收藏功能 | 通过 | 收藏符号颜色改变对应当前状态 |
| TC-FN-007 | 查看我的收藏 | 通过 | 收藏数据符合操作 |
| TC-FN-008 | 取消收藏 | 通过 | 收藏符号颜色改变对应当前状态 |
| TC-FN-009 | 用户登录 | 通过 | 登录流程完整，与未登录页面区分 |
| TC-FN-010 | 用户注册 | 通过 | 数据验证完善，可以登录 |
| TC-FN-011 | 修改密码 | 通过 | 密码修改机制完善，密码成功修改 |

核心功能通过测试，无功能缺陷。新闻列表渲染性能有待提升，应满足部分用户快速浏览需求。

# 五、参考文献

[1] 华为技术有限公司. HarmonyOS应用开发指南[M]. 北京：人民邮电出版社，2023：1-356.

[2] 张明，李华，王刚. 分布式操作系统原理与HarmonyOS实践[J]. 计算机工程与应用，2023，59(15)：1-12.

[3] 陈晓，刘伟等. 移动应用用户体验设计研究[A]. 第十届全国人机交互学术会议论文集[C]. 北京：清华大学出版社，2022：45-52.

[4] 华为开发者联盟. ArkTS语言开发规范与最佳实践[EB/OL]. [https://developer.harmonyos.com](https://developer.harmonyos.com/)，2023.

[5] Johnson M, Brown K, et al. Mobile Application Architecture Design Patterns[M]. New York: O'Reilly Media, 2022：123-156.

[6] 王磊，张静. 基于声明式UI的移动应用开发框架研究[J]. 软件学报，2023，34(8)：1890-1905.

[7] 国家新闻出版署. 移动互联网应用程序信息服务管理规定[S]. 北京：中国标准出版社，2022.

# 六、心得体会

## 1. 综合学习心得

通过本次HarmonyOS应用开发综合设计，深入理解了分布式操作系统的开发理念和技术架构。在项目实践中，掌握了基于ArkTS的声明式开发模式，熟悉了HarmonyOS特有的构建工具和包管理机制。从最初的需求分析到架构设计，再到模块配置和测试验证，每个环节都环环相扣、缺一不可。在配置管理方面，通过module.json5等文件的编写，理解了应用权限、设备兼容性和构建配置的重要性。测试环节的设计与执行让我认识到质量保障的必要性，只有经过充分验证的功能才能真正交付使用。这次经历让我明白，一个成功的项目不仅需要扎实的编程基础，更需要系统性的工程思维和严谨的开发流程，这些收获将对我未来的技术成长产生深远影响。

## 存在问题及分析

在完成本次项目中也遇到了不少问题，例如在实现点击头像更改图片时，由于测试环境为虚拟机测试，导致需要从虚拟机相册中获取图片。要先把本机图片通过device file explorer传入并保存在相册再获取。增加调试代码并不断查看日志后发现，虽然数据库已经获取图片url，并显示图片更新成功，但图片仍不更改。是因为权限管理不够完善，虽然应用已申请了读写媒体文件权限，但缺乏运行时动态权限申请机制，导致在系统版本上仍出现权限拒绝而无法正常选择图片。在aboutToAppear生命周期中，用RdbUtils.queryCollectByAccount来获取收藏列表，然后通过RdbUtils.queryNewsByID获取每个收藏对应的新闻详情。但是，在获取到收藏列表后，使用了Promise.all来等待所有新闻详情查询完成，导致账户信息还没有获取到就开始查询收藏，导致之前点击收藏功能，在我的收藏一直加载不了数据。新闻app首次打开，不显示数据，要通过点击其他按钮才能加载数据，可能原因是刚开始打开app，在数据库里的数据还没加载好，就触发渲染，数据加载好了，需要通过UI改动来唤醒页面数据加载。

## 今后努力的方向

在今后的开发道路上，我将着重提升系统的架构设计与性能优化能力。当前项目虽然实现了基础功能，但在模块解耦和状态管理方面仍有不足，需要引入更成熟的设计模式来增强代码的可维护性。面对性能瓶颈，特别是图片处理与内存管理环节，需深入掌握资源调优技巧，构建完整的监控体系。同时要完善异常处理机制，建立分层级的错误处理策略，确保用户体验的连贯性。在技术广度上，积极探索鸿蒙系统的分布式能力与AI服务，写出更好的创新项目。

# 七、附录

## 1. 核心配置文件清单

### 1.1数据库构建 (RdbUtils.ts)

import relationalStore from '@ohos.data.relationalStore';  
import preferences from '@ohos.data.preferences';  
import { NewsBean } from '../mode/NewsBean';  
import { Users } from '../mode/Users';  
import { Comment } from '../mode/Comment';  
import { CollectBean } from '../mode/CollectBean';  
import { CommentItem } from '../mode/CommentItem';  
  
export default class RdbUtils {  
 private static rdbStore: relationalStore.RdbStore;  
 static dataPreferences: preferences.Preferences | null = null;  
  
 static setStore(store: relationalStore.RdbStore) {  
 RdbUtils.rdbStore = store;  
 //创建数据库表  
 const NEWS\_TABLE = 'CREATE TABLE IF NOT EXISTS NEWS (ID INTEGER PRIMARY KEY AUTOINCREMENT, IMAGE TEXT NOT NULL,TITLE TEXT NOT NULL,MDESC BLOB,TYPE INTEGER,TIME TEXT NOT NULL)';  
 const USERS\_TABLE = 'CREATE TABLE IF NOT EXISTS USERS (ID INTEGER PRIMARY KEY AUTOINCREMENT, ACCOUNT TEXT NOT NULL, PASSWORD TEXT NOT NULL, AVATAR TEXT)';  
 const COMMENT\_TABLE = 'CREATE TABLE IF NOT EXISTS COMMENT (ID INTEGER PRIMARY KEY AUTOINCREMENT, HEAD TEXT NOT NULL,USERNAME TEXT NOT NULL,MDESC BLOB,NID INTEGER,TIME TEXT NOT NULL)';  
 const COLLECT\_TABLE = 'CREATE TABLE IF NOT EXISTS COLLECT (ID INTEGER PRIMARY KEY AUTOINCREMENT, USERNAME TEXT NOT NULL,NID INTEGER,TIME TEXT NOT NULL)';  
  
 this.executeSql(NEWS\_TABLE);  
 this.executeSql(USERS\_TABLE);  
 this.executeSql(COMMENT\_TABLE);  
 this.executeSql(COLLECT\_TABLE);  
  
 // 检查并升级数据库  
 this.upgradeDatabaseIfNeeded().then(() => {  
 this.initData(); //初始化新闻数据  
 });  
 }  
  
 // 添加数据库升级检查  
 static async upgradeDatabaseIfNeeded(): Promise<void> {  
 try {  
 console.info("Checking database structure...");  
  
 // 检查 USERS 表是否有 AVATAR 列  
 const result = await this.getStore().querySql("PRAGMA table\_info(USERS)");  
 let hasAvatarColumn = false;  
 let columns = [];  
  
 while (result.goToNextRow()) {  
 const columnName = result.getString(result.getColumnIndex('name'));  
 const columnType = result.getString(result.getColumnIndex('type'));  
 columns.push(`${columnName} (${columnType})`);  
  
 if (columnName === 'AVATAR') {  
 hasAvatarColumn = true;  
 }  
 }  
  
 console.info("Current USERS table columns: " + columns.join(', '));  
  
 if (!hasAvatarColumn) {  
 console.info("Adding AVATAR column to USERS table...");  
 await this.executeSql('ALTER TABLE USERS ADD COLUMN AVATAR TEXT');  
 console.info("Successfully added AVATAR column");  
 } else {  
 console.info("AVATAR column already exists");  
 }  
  
 } catch (error) {  
 console.error("Database upgrade failed: " + error);  
 }  
 }  
  
 // 添加调试方法查看用户表内容  
 static async debugUsersTable(): Promise<void> {  
 try {  
 console.info("=== DEBUG: USERS TABLE CONTENTS ===");  
 const result = await this.getStore().querySql("SELECT \* FROM USERS");  
 let rowCount = 0;  
 while (result.goToNextRow()) {  
 rowCount++;  
 const id = result.getLong(result.getColumnIndex('ID'));  
 const account = result.getString(result.getColumnIndex('ACCOUNT'));  
 const password = result.getString(result.getColumnIndex('PASSWORD'));  
 let avatar = '';  
 try {  
 avatar = result.getString(result.getColumnIndex('AVATAR')) || '';  
 } catch (e) {  
 avatar = 'NOT\_FOUND';  
 }  
 console.info(`User ${rowCount}: ID=${id}, ACCOUNT=${account}, PASSWORD=${password}, AVATAR=${avatar}`);  
 }  
 console.info(`Total users: ${rowCount}`);  
 console.info("=== DEBUG END ===");  
 } catch (error) {  
 console.error("Debug query failed: " + error);  
 }  
 }  
  
 // 添加更新用户头像  
 static async updateUserAvatar(account: string, avatarPath: string): Promise<number> {  
 let predicates = new relationalStore.RdbPredicates('USERS');  
 predicates.equalTo("account", account);  
  
 const valueBucket = {  
 "AVATAR": avatarPath  
 };  
  
 return RdbUtils.getStore().update(valueBucket, predicates);  
 }  
  
 // 添加获取用户头像  
 static async getUserAvatar(account: string): Promise<string> {  
 let predicates = new relationalStore.RdbPredicates('USERS');  
 predicates.equalTo("account", account);  
  
 return new Promise<string>((resolve, reject) => {  
 RdbUtils.getStore().query(predicates).then((result) => {  
 let avatar = '';  
 while (result.goToNextRow()) {  
 // 获取 AVATAR 字段，如果不存在则返回空字符串  
 try {  
 avatar = result.getString(result.getColumnIndex('AVATAR')) || '';  
 } catch (e) {  
 avatar = '';  
 }  
 }  
 resolve(avatar);  
 }).catch((error) => {  
 reject(error);  
 });  
 });  
 }  
  
 static setPreferences(dataPreferences: preferences.Preferences) {  
 RdbUtils.dataPreferences = dataPreferences;  
 }