江 苏 科 技 大 学

本 科 毕 业 设 计（论文）

学　　院　计算机学院

专　　业　物联网工程（卓越）

学生姓名　张寒宇

班级学号　152210703235

指导教师　房靖、张笑非

二零一九年五月

江苏科技大学本科毕业论文

基于 MQTT 的船联网数据监控 APP 设计与实现

APP Design for Internet of Ships Based on MQTT

江 苏 科 技 大 学

毕业设计（论文）任务书

学院名称：计算机学院　　专　业：物联网工程(卓越)

学生姓名：张寒宇　　学　号：152210703235

指导教师：房靖、张笑非　　职　称：副教授、讲师

2018 年　**12** 月 **28** 日

毕业设计（论文）题目：

基于 MQTT 的船联网数据监控 APP 设计与实现

一、毕业设计（论文）内容及要求（包括原始数据、技术要求、达到的指标和应做的实验等）

MQTT 是物联网技术中的一种机器到机器通信协议，它能够提供基于发布/订阅模式的超轻量消息传输服务。开源项目 Paho 提供了面向 Java 语言的 MQTT 客户端开发库，使得开发者能够设计运行在 JVM 上的物联网应用，并利用 MQTT Broker 完成大规模结点的消息交换。

本课题要求采用自顶向下的设计方法，使用 MQTT 和 Android SDK 完成船联网数据监控 APP 的设计与实现，利用 Paho 提供的 MQTT API 订阅来自船联网的姿态、方位、电力等数据，提供数据分析和可视化功能，能够实时查看船舶的位置、轨迹、船载发送机启停等数据，并利用 Baidu API 提供地图服务。

二、完成后应交的作业（包括各种说明书、图纸等）

1. 毕业设计论文一份;（不少于 1.5 万字）

2. 外文资料翻译一篇;（不少于 5000 英文单词）

3. 船联网数据监控件系统一套

三、完成日期及进度

2019 年 2 月 25 日至 2019 年 6 月 7 日，共 15 周。

进度安排：

2 月 25 日~3 月 10 日(两周)： 系统调研，收集资料，课题设计准备；

3 月 11 日~3 月 24 日（两周）： 系统分析，系统总体设计；

3 月 25 日~5 月 5 日（六周）： 详细设计，编程、调试，系统成型；

5 月 6 日~5 月 12 日（一周）： 论文初稿；

5 月 13 日~5 月 19 日（一周）： 修改论文并定稿；

5 月 20 日~6 月 7 日（三周）： 总结设计过程并完成毕业设计答辩。

四、主要参考资料（包括书刊名称、出版年月等）：

[1] 彭友. 基于船联网的智能船舶协同综合保障平台的设计与实现[D].江苏科技大学,2017.

[2] 程子清.Flutter 开发智慧城市相关 APP 具体实践[J].电子技术与软件工程,2018(19):56-57.

[3] 周勇,程子清.Flutter 的原理深度剖析[J].电脑编程技巧与维护,2018(11):19-21.

[4] 刘劭. 基于微服务的教学支持平台服务端的设计与实现[D].南京大学,2018.

[5] 马雄. 基于微服务架构的系统设计与开发[D].南京邮电大学,2017.

[6] 袁青杰. 智能家居系统扩展及应用服务的技术研究[D].西安电子科技大学,2018.

[7] 白昊. 基于 MQTT 协议的物联网平台设计与实现[D].西安邮电大学,2018.

[8] 金诚.移动应用跨平台开发框架的比较分析[J].民营科技,2018(10):150-152.

[9] 王玉洋. 基于微信小程序的移动学习平台环境构建与系统设计开发[D].南京大学,2018.

[10] 汤汪艳. 基于用户体验的手工艺微信小程序设计研究[D].中国美术学院,2018.

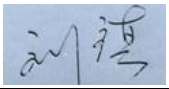
[11] 杨沉. 视觉符号在手机界面设计的应用研究[D].江西师范大学,2016.

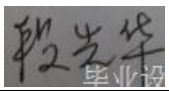
[12] 何浩.基于色彩心理学的 UI 设计研究[J].中国新通信,2015,17(09):11.

[13] 王瑞,兀玉洁,李燕苹.基于 MQTT 协议的物联网实训云平台设计[J].工业控制计算机,2018,31(09):101-103.

[14] 靳良真.船联网的信息安全传输系统研究[J].数字通信世界,2018(07):243.

[15] 钟南. 船联网网络融合关键技术研究与仿真[A]. 中国科学技术协会、交通运输部、中国工程院.2018 世界交通运输大会论文集[C].中国科学技术协会、交通运输部、中国工程院:中国公路学会,2018:8.

系(教研室)主任：  （签章） 2019 年 1 月 17 日

学院主管领导：  （签章） 2019 年 1 月 17 日

摘　要

在物联网产业蓬勃发展的当下，物物互联的思想也在不断融入船舶信息服务领域，形成了船舶间的互联互通的船联网。船联网的出现极大的提升船舶行业的用户体验，而在船联网系统中，APP 是最直接与用户接触的方式。船联网 APP 通过对数据的解析、可视化处理让用户更加直观地了解系统的状态。因此，本文致力于设计出可以稳定、可靠、跨平台工作的船联网 APP。用户可以在任何地方以任何终端对船联网系统进行交互，从而保障船舶安全，提升运输效率。

本设计聚焦于用户与船联网系统的交互媒介——移动端 APP。在系统中，APP 通过 MQTT 协议接入传感网，获取来自传感节点的电力、航线、姿态数据。获取电力数据后，APP 以折线图的形式对数据进行展示。获取航线数据后，APP 通过高德地图绘制航线轨迹，展示船舶位置。获取姿态数据后，展示姿态信息，并通过 3D 模型模拟船舶姿态。APP 对单纯的数据进行可视化处理，使用户可以更直观了解船联网状态，提高用户体验。

除功能性设计外，本设计根据现有系统的不足之处做出以下创新：首先，利用跨平台开发框架 Flutter 实现一次开发、多端复用。其次，考虑到船上复杂的网络环境，本设计使用轻量、开源、易于实现的 MQTT 协议保障数据传输的安全性与可靠性。最后，本设计从色彩心理学的角度出发，重新设计 UI 与交互，以情感化的 UI 设计令 APP 与用户产生共鸣，在满足多端跨平台协同处理的同时提升用户的操作体验。

关键词： 船联网；Flutter；Spring Boot；MQTT；色彩心理学

Abstract

With the rapid development of the Internet of Things (IOT) industry, the idea of interconnection of things is constantly integrated into the field of ship information service, forming the interconnection of ships. The emergence of ship networking greatly improves the user experience of the ship industry, and in the ship networking system, APP is the most direct way to contact users. Through data analysis and visualization, APP enables users to understand the state of the system more intuitively. Therefore, this paper is devoted to the design of stable, reliable and cross-platform ship network APP. Users can interact with the ship networking system at any terminal anywhere, so as to ensure the safety of ships and improve transportation efficiency.

This design focuses on the interactive medium of user and ship networking system - Mobile APP. In the system, APP accesses the sensor network through MQTT protocol to obtain the power, route and attitude data from the sensor nodes. After obtaining the power data, APP displays the data in the form of a broken line graph. After acquiring the route data, APP draws the route trajectory and shows the position of the ship through the Gaud map. After obtaining the attitude data, the attitude information is displayed and the ship attitude is simulated by 3D model. APP visualizes the simple data so that users can more intuitively understand the status of ship networking and improve user experience.

In addition to functional design, this design makes the following innovations according to the shortcomings of the existing system: First, the cross-platform development framework Flutter is used to realize one-time development and multi-end reuse. Secondly, considering the complex network environment on board, this design uses lightweight, open source and easy-to-implement MQTT protocol to ensure the security and reliability of data transmission. Finally, from the perspective of color psychology, this design redesigns the UI and interaction. The emotional UI design makes APP resonate with users, and improves the user's operating experience while satisfying multi-terminal and cross-platform collaborative processing.

Keywords: Internet of Ships; Flutter; Spring Boot; MQTT; color psychology

目　录

第一章 绪论

1.1 研究背景及意义

从数百年前哥伦布发现新大陆今，航运的最大的问题已不再是“如何达到”，而是保障航运的过程更安全、高效、环保。为保障船舶安全出行，保护环境生态健康，“十二五”期间，交通部提出了“现代交通运输业要靠信息化来带动，没有信息化，就没有现代化”的发展方向，决心推进智慧交通发展，加强信息化建设。

根据国家物联网产业发展规划，以绿色、畅通、平安、高效的内河航运作为发展要求，为提高我国内河船联网航运信息服务水平，国家发改委批准了物联网应用示范工程——船联网项目。随着 2013 年在上海正式启动，国内航运物流水准日趋国际化，船联网也将会是未来内河航运信息化、智能化发展的趋势。

船联网的发展也离不开 5G 的支持，在过去，由于网络制式与延时等问题，船联网的实时性仍存在问题。但随着技术的发展，2018 年，中国 IMT—2020（5G）推进组发布了 5G 研发试验第三阶段第一批规范。规范表示，5G 相比传统制式网络，速率上可提升 10-100 倍，连接密度可提升 10 倍，时延可降为 4G 的 1/5 以下。虽然在下载传输等方面，网络延迟差距很难感觉出来，但在船联网数据监控等要求毫秒级的互动操作响应的领域，5G 将极大的提高船联网的实时性与准确性。

5G 的出现保障了船联网领域的实效性与可靠性，船联网将进入一个新的时代。在新的船联网领域，船与船、船与航道、船与人互联互通，将原本充满危险、不确定的航程，逐渐变得智能、可控。新一代的物联网系统，将以“硬件模块化、功能软件化、平台虚拟化”为核心，真正实现船岸智能信息一体化。

1.2 研究现状

1.2.1 国内研究现状

我国船联网主要分布在南方。南方各省根据各自的自然环境、经济社会，展开了具有当地特色的船联网建设活动。

以长江三角洲水系为例，江苏省设计出台了“江苏省船联网示范工程”。江苏利用物联网和现代通信技术，结合与挖掘水路交通信息，建立了智能化、数字化的航运信息服务体系[1]。江苏省的船联网服务体系和水路运输管理部门密切配合，为内河运输业提供

了全方位的信息服务。

除了长江三角洲之外，水网密布的湖北是承接南北运输的关键地区。2015 年，湖北水运物流发展座谈会透露，湖北将在未来 5 年大力打造类似于“滴滴快车”的“船联网”。届时，公众可在船联网平台查询港口、装卸、船货以及船员的相关信息。全省水运体系通过接入此体系实现信息共享，以此推进湖北航运业发展。

除了内河航运，我国近海航运也有一定规模的船联网。福建由于自身渔业发达，更多地将船联网技术应用于渔船监管。此外，由于毗邻台湾海峡，地缘环境特殊，海况复杂，导致福建渔业事故频发。为此，福建省将北斗卫星导航系统(BeiDou Navigation Satellite System)引入船联网系统。为渔船提供位置监测、跨区域信息共享、海上通话、求助预警、救援辅助、指挥调度、出入港检测、船舶资料管理、航程记录、航次统计等服务。有效保障渔民的出海安全，推动了当地渔业发展。

1.2.2 国外研究现状

2017 挪威国际海事展上，日本邮船 NYK 与挪威船级社 DNV GL 发布了海事数据中心研究成果。自项目启动以来，日本邮船旗下 4 艘集装箱船持续将数据传输至挪威船级社的 Veracity 数据平台。双方基于这些运营数据，实现了船舶性能监测与维护检验等功能。挪威船级社 DNV GL 还预测了船联网的发展趋势。一个是数字化趋势，利用数据模型来描述船舶及其系统。还有“基于状态”的趋势，利用持续采集的数据来调整船舶运营和进行预测性检验。

展会 5 月 31 日，日本邮船 NYK 与挪威海事信息技术企业 Dualog 签订了战略合作协议。根据协议内容，日本邮船与挪威海事将共同开发新一代船联网技术。该项目包含三个阶段，第一阶段：数据中心基本架构，包括如何实现数据收集、采集后的数据管理等内容；第二阶段：对数据质量、访问权限及安全性进行研究，以完成预测性维护和船舶监测等功能；第三阶段：建立全新的数字化商业模式。

在美国，航运主要港口如休斯顿港，纽约港，迈阿密港。通常使用射频识别技术作为船联网技术的基础，确保物流的透明度和打击恐怖主义。使用 RFID 标签确定港口物品的确切位置，从而实现快速定位。通过船联网系统，能够快速定位各个端口的集装箱，卡车和船舶。所有设备的数据汇总进行数据挖掘和集成数据，以便进行决策和性能优化。同时基于船联网的其他技术，如 GPS，可以跟踪集装箱内外的集装箱运动，确保其货物运输的安全性。同时，通过网络监控精确，保障了船舶运营的可视性和透明度。

1.3 论文研究方向

我国的船联网部署工作较日本、欧美等地区起步较晚。如今虽已初具规模，但与船联网系统相匹配的服务和信息管理仍处于较低水平。这致使船舶信息采集成本过高，大大降低了航运效率，对船联网行业的发展造成了不利影响。

本系统针对船联网在信息管理与服务上面的不足之处，提出针对性的解决方案，利用低功耗传感网对信息进行采集，可靠的传输协议保证了船联网数据传输过程中的准确与实时。

在 APP 方面，利用当下最新的 Flutter 技术实现多端跨平台协同交互，保证在 iOS，Android，Windows，macOS，Linux，web 多平台对船联网系统有一致性的操作。同时对单一数据进行可视化处理，使数据交互更为直观具体，方便对船联网系统的监控与维护。

同时，船联网收集的大量数据在 5G 与机器学习的催化下，也将产生巨大的价值，更多的管理、监督和服务决策将基于大数据分析而得出更准确的结果。构建船联网信息可视化平台，保障用户多端协同处理，利用机器学习与云计算进行模型训练，保障船舶行驶安全，船内通信畅通，智能控制船舶电力，达到节能减排、降低成本、提高效能。这才是新一代船联网的发展方向。

第二章 理论基础与相关技术

2.1 Flutter 简介

Flutter 本身支持 Android 和 iOS 两个平台，除了性能和开发语言上的“native”化之外，它还提供了两套设计语言的控件实现 Material & Cupertino，可以帮助 APP 更好地在不同平台上提供原生的用户体验 [2]。

Flutter 是由 Google 推出的跨平台 UI 框架。Flutter 底层基于 C, C++, Dart runtime，使用 Skia2D 作为渲染引擎。相较于传统 Hybrid APP，Flutter 使用了一种全新的思路。Flutter 是一套重新设计了 UI 控件、渲染逻辑甚至 runtime 的跨平台的 UI 框架。其渲染引擎依靠由 C++实现的 Skia2D 图形库来实现。在渲染中，依赖原生环境的仅有图形绘制相关的接口，这最大程度上保障了不同平台下体验一致性。逻辑处理使用支持 AOT 与 Hot reload 的 Dart 语言，效率也较 JavaScript 高许多。由于以上特性，Flutter 在运行时拥有媲美 Native APP 的性能。

Flutter 的架构主要包括三个部分:Framework，Engine 和 Embedder，如图 2-1。Framework 包括 Android(Material Design)风格和 iOS(Cupertino)风格的 Widgets。Framework 层主要由 dart 实现，包含了 Text、Image、Button 按钮等基础 Widgets，也包含了渲染，动画，手势等交互操作。

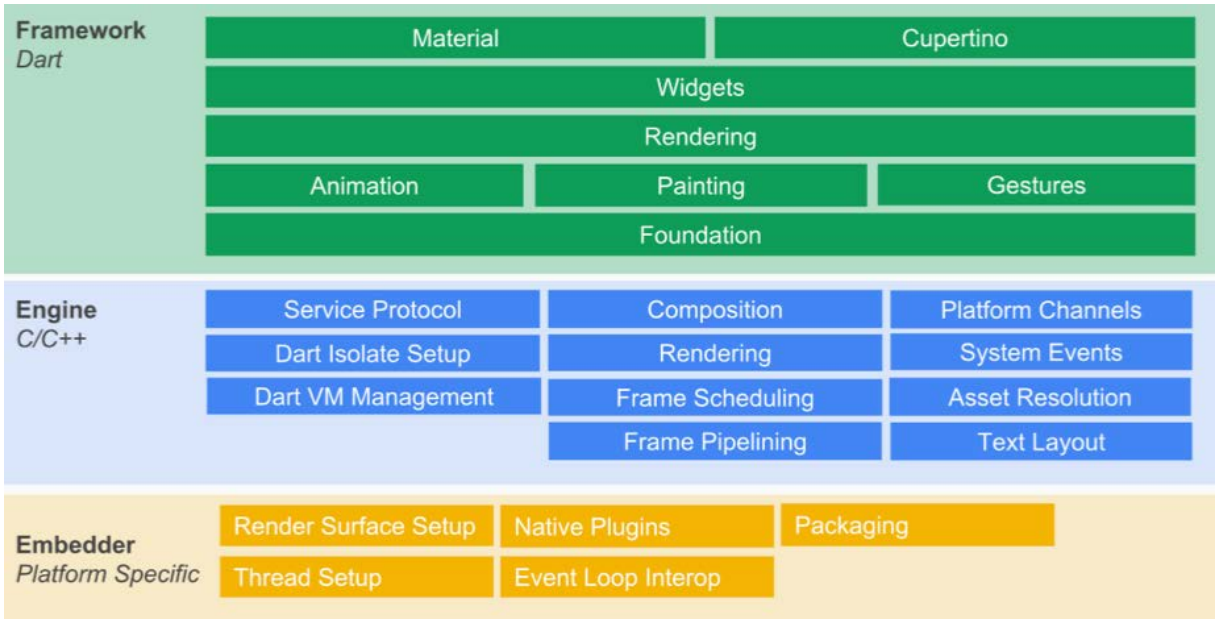


图 2-1：Flutter 框架结构

Engine 层主要包括:Skia, Dart runtime 和 Text，由 C++实现。Skia 是 Googole 开源的 2D 图形库，提供了多种不同软硬件平台的通用 API。其已作为 Google Chrome, Firefox 等产品的图形引擎。Skia 同样提供了 Windows, macOS, iOS, Android, Linux 等平台的支持。Dart 部分包括:Dart 运行时环境(Dart Runtime)，垃圾收集器。Debug 模式下，还提供了 JIT(Just In Time)支持。Release 或 Profile 模式下，不存在 JIT 部分，而是预编译(AOT)生成了原生的 arm 代码。Text 部分即文本渲染，Text 渲染层次依次为 libtxt(字体选择，分隔行)、HartBuzz（用于字形选择和成型）、Skia 或 GPU 作渲染（在 Android 上使用 FreeType，iOS 上使用 CoreGraphics）。

Embedder 层用于嵌入，即把 Flutter 嵌入其他平台。嵌入工作主要包括渲染 Surface，线程设置，平台插件等。Flutter 与平台的相关度很低，平台仅提供一个画布(Canvas)，渲染工作与逻辑处理都在 Flutter 内部，这保证了 Flutter 拥有良好的跨端一致性。

2.1.1 Framework 层架构

Flutter 的 Framework 完全由 Dart 实现，有着清晰的分层架构如图 2-2 所示。这种分层架构除了可以直接使用 Flutter 的 Widget 控件外，还可以修改每一层实现，便于的定义属于自己的空间。Framework 依赖 Engine 层，Engine 引擎主要包括图形绘制引擎（Skia）、文字排版引擎（libtxt）和 Dart 运行时环境。Engine 全部使用 C++实现，这保证 Framework 可以用 Dart 调用 Engine 的强大能力。

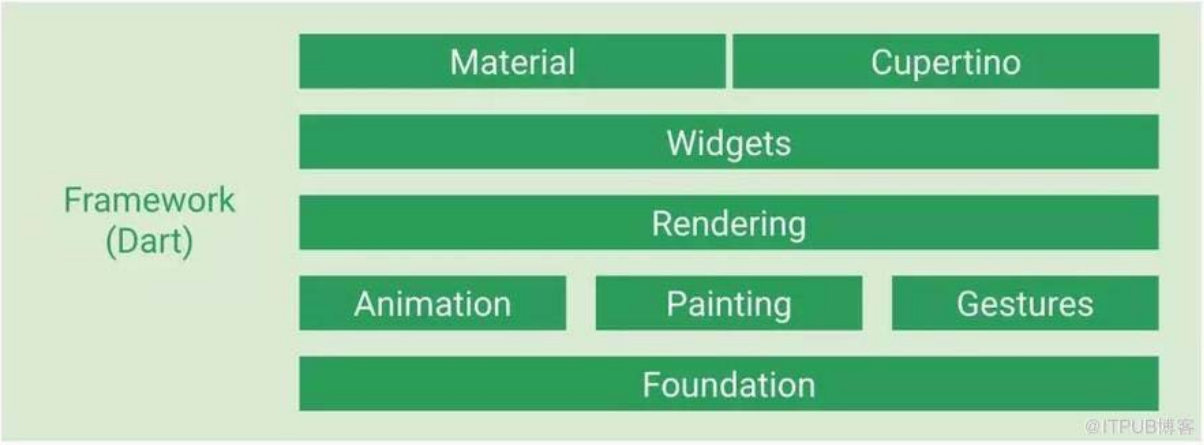


图 2-2：Framework 层架构

Framework 最底层为 Foundation，包含基础的、提供给其他层工具类和方法。Painting(绘制层)封装了 Engine 层提供的绘制接口，可以在绘制控件或图形时提供更方便的接口。如缩放后的位图、文本、生成阴影以及在 Box 周围绘制边框等。Animation

层封装了动画相关类，提供丰富的内置插值器。Gesture 封装了手势识别相关的功能，如触摸事件和手势识别。

2.1.2 Flutter 渲染库（Rendering）

Flutter 的 Widget Tree 在显示时会转换成对应的 RenderObject(渲染对象)来实现布局和绘制。一般只会在调整布局，或者需要自定义控件变化实现某些特殊功能时，才需要考虑 RenderObject Tree 的细节。Flutter 视图树包含了三种树 Widget, Element 和 RenderObject。

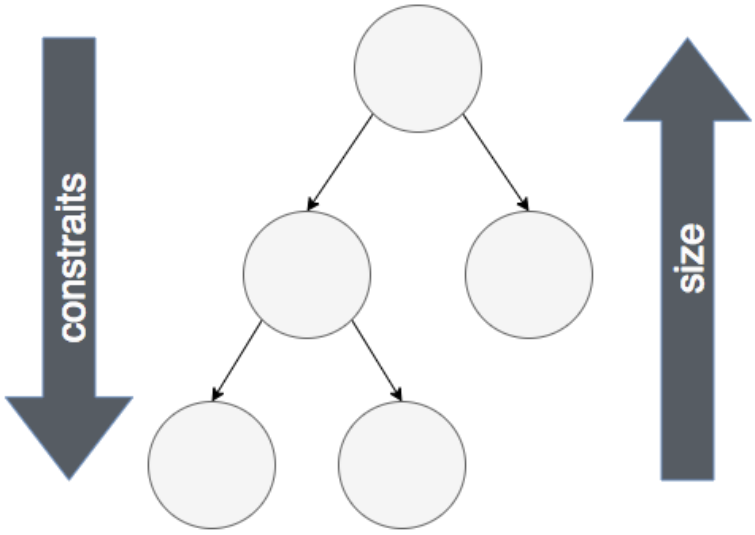


图 2-3：RenderObject 树结构

RenderObject 树中的每个对象在布局中都接受父对象的 Constraints 数据，决定自己的大小。然后父对象按照自己的逻辑布局每个子对象的位置。

子对象不存储自己在容器中的位置，所以在它的位置发生改变时并不需要重新布局或者绘制。子对象将位置信息存储在自己的 parentData 对象中，parentData 对象由子对象的父对象负责维护。得益于这种简单的布局逻辑，某些 Flutter 内的节点可以设置 Relayout boundary（布局边界），如图 2-4，当边界内对象出发重布局时，不会对边界外的对象产生影响，反之亦然。

布局完成后，RenderObject 树中的节点对象都有了明确的尺寸和位置。此时，Flutter 会将所有对象通过 Skia 绘制到不同的图层上。

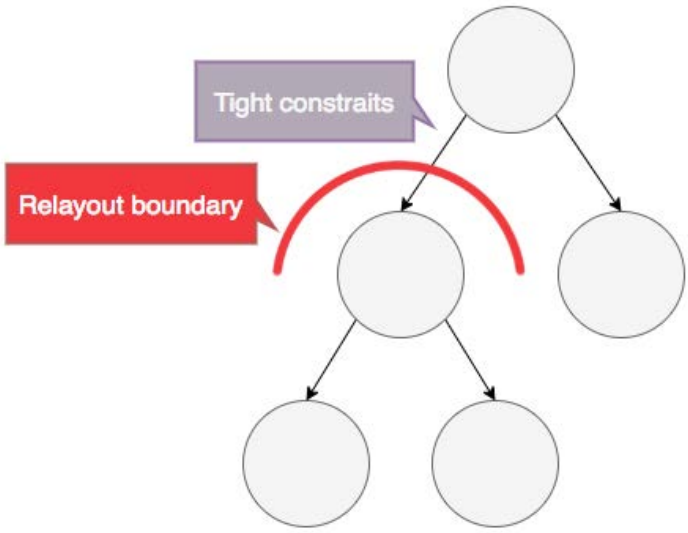


图 2-4：RenderObject 树中的 Relayout boundary（布局边界）

在绘制节点时，Flutter 会对 RenderObject 树进行深度遍历。当出现如图 2-5 所示情况，第 2 节点与绘制它的背景和前景绘制在不同的图层上。因为第四个节点进行了切换图层动作（比如视频），而第 6 节点也一同绘制在红色图层上。这样导致第二个节点的前景（第 5 节点）需要部分重绘时，逻辑上与它毫不相干但处于红色图层的第 6 节点也必须重绘。

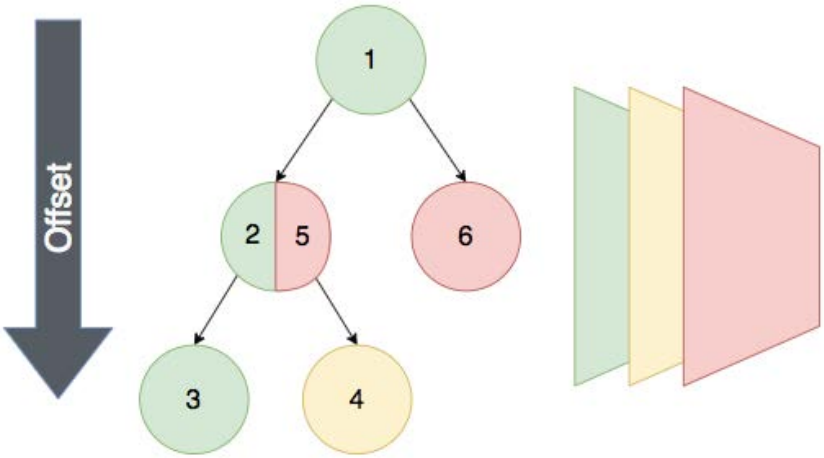


图 2-5：RenderObject 树中的进行切换图层

为了避免上述情况，Flutter 提供了 Repaint boundary（重绘边界）概念，如图 2-6 所示。在进入和走出重绘边界时，节点会被强制切换至新的图层，以避免边界内外的互相影响。典型的应用场景就是 ScrollView，当滚动内容重绘时，一般情况下其他内容是不需要重绘的。虽然可以手动设置重绘边界，但 Flutter 为需要用到重绘边界的常用控件默认设置了范围，所以一般不需手动设置 Repaint boundary[3]。

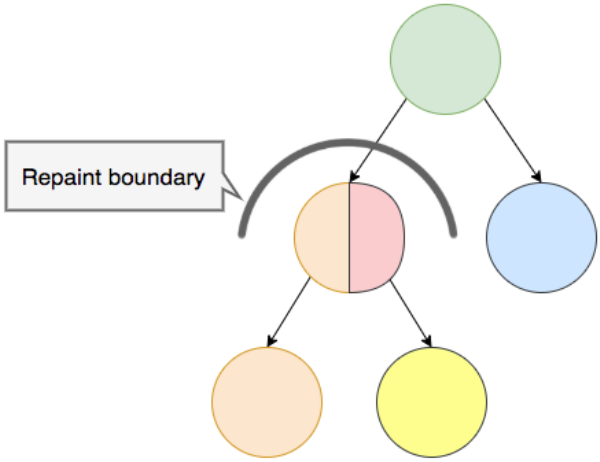


图 2-6：RenderObject 树中的 Repaint boundary（重绘边界）

2.2 Spring Boot

Spring 是一个轻量级控制反转(IOC)和面向切面(AOP)的容器框架。随着 Spring 的运用越来越广泛以及越来越多功能与 Spring 整合，大量的 XML 配置和依赖管理也让 Spring IO 平台饱受非议。在 2013 年 SpringOne 2GX 会议上，Pivotal 的 CTO Adrian Colyer 回应了这些批评，并提出 Spring IO 的目标之一就是实现免 XML 配置的开发体验 [4]。

Spring Boot 并不是要成为 Spring IO 平台中众多"Foundation"层的替代者，而是为了简化 Spring 框架的开发过程中配置、调试、部署等工作的工具。Spring Boot 集成了大量在开发过程中会用到的基础框架。如 Web 容器（Tomcat、Jetty）、日志框架（log4j、logback）、JMS 框架、持久化框架（Mybatis、JPA）、SQL（MySQL、Oracle）、NoSQL（Redis、MongoDB）、缓存框架等，如图 2-7。

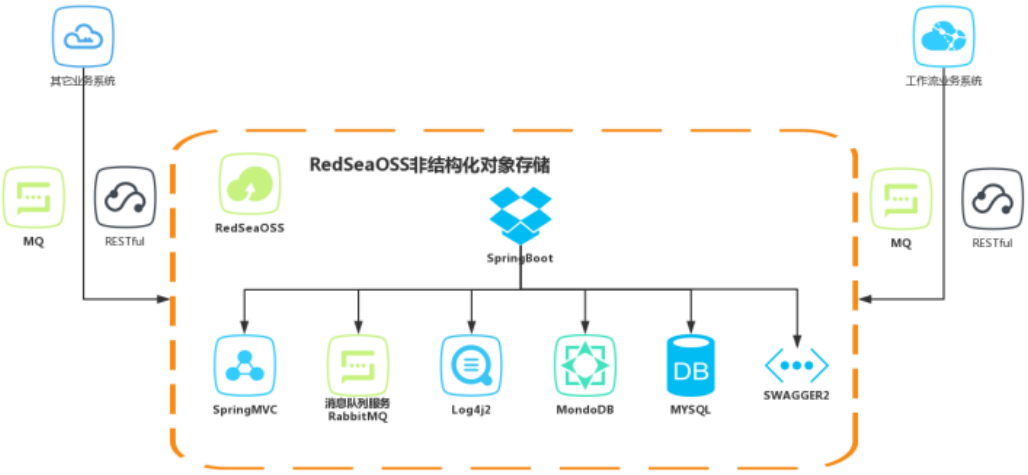


图 2-7：Spring Boot 与常用基础框架

使用 Spring Boot 开发过程中，仅需在 Maven、Gradle 中添加少量 Spring Boot starter，即可在系统中集成所需的框架。让原本复杂的 XML 配置大大简化，同时也解决了不同框架之间的依赖冲突，便于系统的升级与维护 [5]。

Spring Boot 主要有这些核心特点：可将项目编译为可执行 jar 包。jar 包含了运行时所需的一切，包括 Web 容器等，便于快速部署。采用完美遵循约定大于配置的理念，自动化配置模块。将各种框架以 starter 的方式配置，简化开发过程，降低配置难度。

2.3 MQTT 协议

MQTT（Message Queuing Telemetry Transport，消息队列遥测传输协议）由 IBM 在 1999 年发布，是一种基于发布/订阅（publish/subscribe）模式的"轻量级"通讯协议，如图 2-8。MQTT 协议是目前物联网的最主要的传输协议之一，它基于 TCP/IP 协议。最大优点在于，MQTT 可以用极少的报文和有限的带宽，为接入设备提供实时、且具有多种可靠性的消息服务。MQTT 作为一种轻量、低带宽的实时通讯协议，在物联网、嵌入式、移动开发等方面有较广泛的应用 [6]。

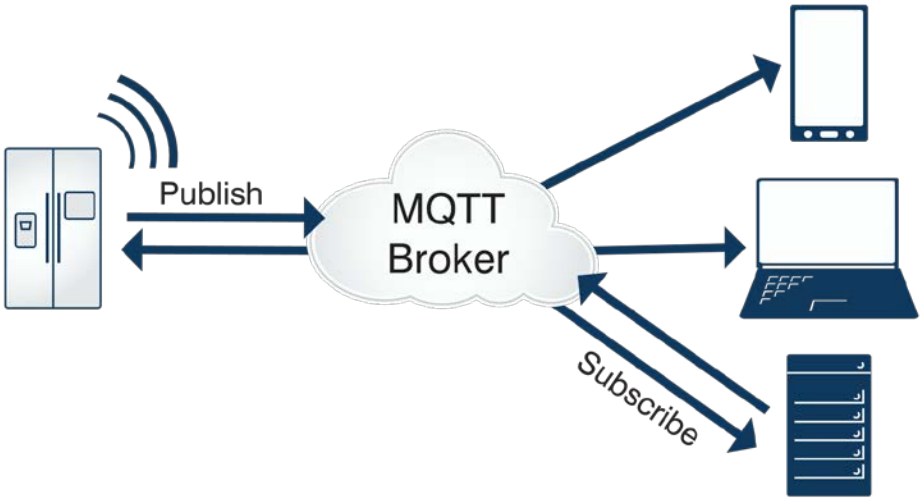


图 2-8：MQTT 的发布/订阅模式

MQTT 是在设计之初就致力于轻量、简单。开源和易于实现的特性使它适用范围非常广泛。在很多网络条件受限的环境中更是如鱼得水，如人机通信（M2M）和物联网。目前在卫星链路通信、偶尔联网的医疗设备、智能家居、及嵌入式设备中已广泛使用 [7]。

由于设计初衷为轻量简单，MQTT 在设计时遵循以下原则：

(1) 精简，删去了可有可无的功能；

(2) 根据传输角色不同，传输拥有两种模式：发布/订阅，方便消息在不同角色传感

器之间传递，解除应用程序耦合；

(3) 允许动态创建主题，降低用户运维成本；

(4) 使用遗言机制，利用 Last Will 通知各方客户端出现异常时中断的机制。

(5) 考虑到复杂的网络环境，在设计之初将低带宽、高延迟、不稳定考虑在内；

(6) 支持长时间的连续会话控制；

(7) 考虑到客户端的计算能力，尽可能降低解析复杂度，头部固定长度 2 字节，最小化交换协议以降低网络流量，提高传输效率；

(8) 根据需求不同，可提供三种不同质量的 QoS. 包括至多一次，至少一次和只有一次三个级别。

(9) 在数据不可知的情况下，对传输数据的类型与格式不做强制要求，保证灵活性。

第三章 系统分析

3.1 投资效益及社会因素分析

3.1.1 经济效益分析

船联网的意义不仅是为了将物联网中现代化的技术应用于传统的内河航运体系，建立实时、准确、高效的航运系统体系，还为了以科学化的管理方式和智能的能效控制来达到降低运营成本，提高运营效率的目的。

在船联网未普及时，内河航运智能化水平远落后其它运输方式。在传统内河航运中，以人为干预作为主要实现方法，粗放式的管理导致了内河航运在规划之初就缺乏准确的数据支撑。粗略的数据监测导致对大数量的船舶无法进行有效地引导。

而整合物联网技术后，船联网系统可以动态捕捉运输船舶，实时获取船舶分布，并对其身份进行识别，准确预测航道内船舶流量，进行分流导流。减少航道拥塞时间，提升单位时间内航道的通行量，提升船舶的航行效率。

除提升船舶航行效率，船联网还致力于打造绿色环保的水运服务。先天上，水运较其他运输方式来说是一种比节能的运输方式。如何进一步提升内河运输环节能耗比，实现节能减排是内河航运的下一个发展趋势。其中，减少停靠次数是一种有效的节能手段。以目前江苏省京杭运河苏北段全线船闸对安装 GPS 的船舶实行“一次缴费、全程服务、无缝调度”的效果来看。虽然船舶马力大小不同，但平均每只船舶单次过闸即可节约 30-60 公斤燃油，通行全段 9 道闸可节省燃油 300-500 公斤，每只船队每次航行可节约成本约 3000 元。

通过普及船联网技术，航行实现远程登记、自动缴费、远程调度。对船舶的非接触式检查，海事部门可节约大量工作时间，减少工作任务。同时船舶停靠次数的减少，也将极大减少能源的消耗，降低碳排放，实现减低运营成本，提高运营效率的目的。

3.1.2 社会因素分析

物联网与 5G 的出现为社会带来新的变革，中国在 5G 通信技术方面也走在了世界前列。物联网产业与 5G 通讯技术即将上升为高新科技领域的领军产业。国家地方也高度重视，对于物联网与 5G 的建设给予了超常规的关注。船联网应用也必将由闭环应用走向开环应用。我国在船联网建设过程中也同时注重引导和推动国产芯片技术。各地方

也出台了关于船联网标准体系全局化、船联网技术产业化的相关文件。未来，中国也将持续推进船联网与相关产业链的发展。

本系统与其相关衍生系统，如移动端 APP，也都遵循国家法律相关规定。系统不会对用户行为进行管理监控，也不会获取如用户密码等受法律保护的信息。且系统不会对用户进行的具体操作，仅仅保留用户访问记录与实践，方便用户核对，或配合相关部门进行法律调查。

船联网系统相关技术栈也遵循各大开源协议使用规范。移动端开发框架 Flutter 授权使用 MIT license 协议，软件可作商业使用，可以修改源码，可以出售相关衍生产品。服务端使用 Spring Boot，Tomcat，遵循 Apache License,2.0 协议，可以修改代码满足需要并作为商业产品发售。数据库 MySQL，遵循 GPL 协议，要求使用 GPL 的软件产品也必须使用 GPL 协议。均按照协议要求使用，秉承开源精神，尊重作者知识产权，维护产品权益。

3.2 技术可行性分析

本系统相较其他系统最大的特点是可以在多平台协同交互，在不同平台达到相同的用户体验。为解决跨平台开发，曾衍生除了许多跨平台的解决方案，然而这些解决方案或由于发布时间过早，时代的局限性型导致其无法再当今移动平台更进一步，如 QT。或由于技术的局限性，导致性能与体验的降低，如结合浏览器相关技术（Html,JS,CSS）与 Native 的 Hybrid APP（混合模式移动应用），如 React Native，Cordova，Weex，Framework7，MUI 之类的框架。Hybrid 框架虽为了解决跨平台而出现，但是不同平台对 JS 的解析有较大区别。iOS 的 JSCore，Android 的 Chrome V8，不同的 JS 的引擎差别很大，导致利用 Hybrid 在开发跨平台应用时需要针对不同平台进行特定优化，不仅影响了用户体验，还提高了开发的成本。

Flutter 的出现为跨平台开发带来转机，与传统 Hybrid APP 不同，Flutter 本质并不是对 Web 进行解析，而是完全使用开源图形库 Skia 作为渲染。渲染过程中，依赖原生环境的仅有图形绘制相关的接口，这最大程度上保障了不同平台下体验一致性。同时使用 C++与系统资源进行交互，相较 JS 大大减少了与原生系统的通信次数，提升了系统资源利用率。 利用 Platform Channels 与系统通信拓展了许多 JS 无法实现的功能，如本系统将会使用到的 TCP 直连 MQTT。

同时 Dart 语言本身在对 AOT 与 Hot reload 的支持，让 Flutter 应用在 debug 模式下

可以对修改的代码进行热重载，即时查看修改效果，极大地提升了开发的效率。"Write once, run anywhere."的设计原则，媲美 Native APP 的性能，丰富的功能拓展与茁壮发展的社区环境都标志着 Flutter 的设计理念的优秀。Flutter 的将会是跨平台领域极为重要的成果，也让同多端协同应用成为可能 [8]。

3.3 系统需求分析

船联网作为一个节点众多的大型交互系统，其 APP 在设计时不仅需要考虑到用户的诉求，亦需要考虑 APP 与其他模块的通信与交互，因此一个好的基础架构对于未来的拓展极为重要。良好的系统架构在设计之初会考虑到使用者，未来发展趋势，平台特点，网络制式，可拓展性等等。因此，要求一个优秀的系统架构应具有以下特点：

运行的可靠性是船联网系统的第一要求，是保障船舶安全航行和业务稳定开展的基础。系统能否在复杂状况下，如输入错误、网络过载、恶意攻击时，保证不死机、不崩溃是首要需要考虑的。因此，在船联网系统设计时应考虑到数据合法性校验和并发过大等特殊情况，对请求数据进行拦截，设置最大并发数，在最糟的情况下以系统不崩溃作为第一原则。

良好的可拓展性，船联网平台的需求并不是一成不变的。随着技术不断发展，新式船舶与通信技术也会不断更迭，此时需要对系统的功能进行拓展，使其在新时代能继续为船舶航行提供保障。故船联网平台在设计时要考虑到遵循可拓展性原则，如面向接口编程，高内聚低耦合等设计原则，为后期拓展留下空间。

系统在设计，开发，测试时应遵循统一的开发守则以保证系统的规范性，如本系统遵循的阿里 Java 代码规范标准、闲鱼 Flutter 开发守则等。规范性的提高有助于系统的未来发展，遵循规范的系统会更易读和更易维护，可提高团队的工作效率。同时规范的遵守保障在开发过程中能够提高代码复用性和降低代码耦合度，在出现问题时能够更及时准确地寻找解决方案。

3.3.1 平台需求

在船联网平台建设过程中，智能感知平台将是一个重要的部分，航运船舶通过安装智能终端将采集数据传输至布设岸基的信息采集设备。结合移动通信、互联网、物联网等通讯方式实现船舶信息的采集与传输。在此过程中如何对数据进行汇总和处理成为无法回避的问题，众多的采集器有着不同的任务，其中大部分仅为采集作用，不需要对数

据进行处理。相较于大批传感采集节点，APP 接收端的数量显得略微单薄。针对这种多采集，少接收的传输特点，并为了保证传输过程中数据的稳定性与实时性。对于监控平台与船联网 APP 之间使用 MQTT 协议进行交互，QoS Level 2，保证传输过程中数据不会重复，不会有脏数据的存在，数据可以准确实时的传递至 APP 端。

3.3.2 用户需求

在整个船联网系统中，APP 端是与用户联系最紧密的一个节点，在复杂的传感网与运算逻辑的基础上，APP 端需要最直观、最有效地与用户进行交互。船联网 APP 端完全是为了解决用户需求，用户不需要对大量庞杂的数据进行接触，就可以直观具体的了解整个船舶系统的运行状态，这样船联网系统才具有实际意义。因此，对于船联网 APP 来说，研究用户的需求是非常必要的。船联网 APP 是为了解决用户在与船联网交互过程中，面对大量数据，可以迅速直观地了解全局的问题。

从用户行为分析，用户在使用船联网系统时主要有以下行为，如图 3-1：

登录/注册账户，用户可登录系统获取与之相关的信息。

船舶状态监测，用户可以通过 APP 对船联网数据进行查看，可以直观地了解到船舶的航行状态，如当前船舶上电力状况，燃料剩余，当前的航行位置与历史航行轨迹，船体姿态数据等。

用户信息管理，用户可根据个人使用习惯对个人信息进行修改，进行定制化服务。

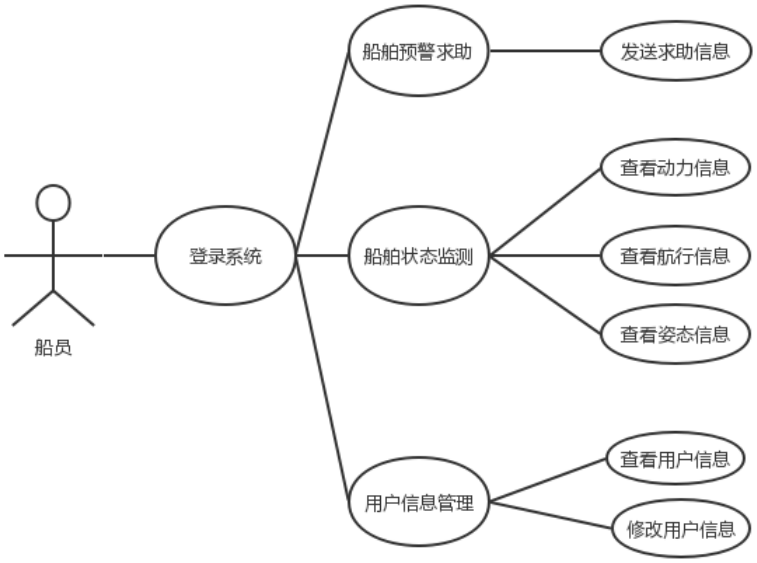


图 3-1：系统用例图

3.3.3 业务需求

船联网系统旨在保证船舶安全出行，获取船舶实时状态，共享水域航运信息，使船舶航运实现绿色高效的目的。因此，船联网 APP 系统在设计时考虑以下需求，如图 3-2：

(1) 安全通讯：APP 在接入船联网中后，可接受来自监管中心的消息推送，对船只发送信息或下达指令。

(2) 电力监控：APP 可通过 MQTT 接入传感网，获取船舶航行时的电力数据，对船舶低电力状态进行监测，保障船舶出行。

(3) 位置监控：APP 可通过 MQTT 接入传感网，获取船舶的位置、速度和方向等信息，并将船舶位置绘制于地图上，历史数据汇总后可形成航行轨迹。监控中心可通过航行数据对所有船只进行实时监管，实现如避碰功能，航道拥塞控制功能，智能航线分析功能等。

(4) 姿态监控：APP 可通过 MQTT 接入传感网，获取船舶姿态信息，并将数据进行可视化处理，使用户可以直观感受船舶姿态，并在船舶姿态不正常时进行预警。

(5) 水文气象预警：当遭遇恶劣自然灾害，如遭遇洪峰、台风和大雾等不利于航行条件的天气时，控制中心可向船舶推送自然灾害预警，提醒船舶提前做好应对，避免不必要的经济损失。

(6) 用户中心：用户可在用户中心查看自己的信息，对信息进行维护或修改，定制个性化服务。

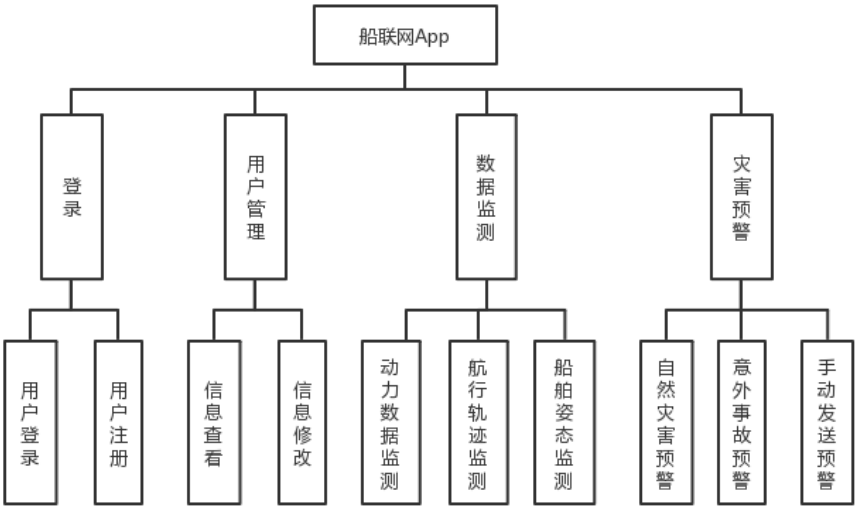


图 3-2：功能结构图

第四章 系统设计

本章主要从系统的总体架构、移动端设计、后端设计、数据库、UI 方面出发，介绍了各个功能模块的设计思路及解决方案。本系统移动端使用 Flutter 框架，后端采用 Spring Boot 框架与动态服务器 Tomcat，数据库为 MySQL，服务器使用腾讯云 CentOS，Docker 容器部署。下文将对船联网 APP 的设计进行详细介绍。

4.1 架构设计

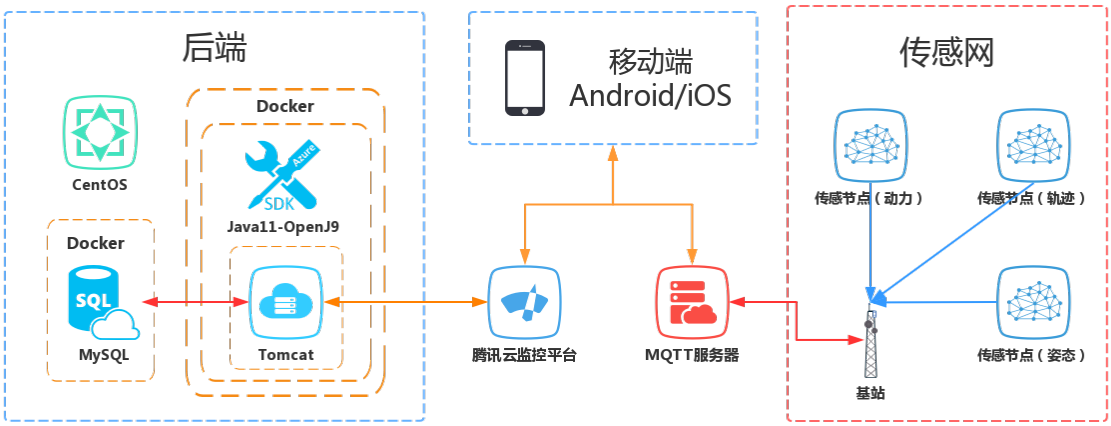


图 4-1：系统架构图

如上图 4-1 所示，船联网 APP 系统主要包括三个部分，除移动端外，还需要后端与 MQTT 的配合。移动端使用 Flutter 作为开发框架，Material Design 为 UI 设计风格。移动端使用 Dio 库与后端建立异步 Http 连接，使用 mqtt_client 对接传感网，获取传感器信息。后端部署使用 Docker，保证环境的独立性，避免不同环境可能产生的影响。后端框架采用 Spring Boot，依赖 Java11-OpenJ9 与 Tomcat9，使用 Mybatis 作为持久层框架，数据池采用 Druid 与 MySQL 数据库进行连接 [9]。

移动端为用户提供统一的 UI 交互界面，向服务器发送请求，进行登录验证等操作。利用 mqtt_client 库与 MQTT 服务器建立连接，接收传感器数据，同时对抽象的数据进行可视化处理，如绘制电力曲线，绘制路线轨迹等。

后端服务器：后端采用 Spring Boot 作为框架，使用了 spring-mvc，mysql-connector，mybatis 等启动器。spring-mvc 接收服务端的业务请求，根据请求地址调用不同 Controller 层方法，controller 会调用 service 层进行业务逻辑处理。需要与数据库通信时，调用持

久层框架 Mybatis，并使用数据池 Druid 与 MySQL 数据库建立连接。

数据库：数据用户存储用户信息，后台数据接收请求与数据库数据校验，返回请求结果。系统数据通过数据库存储，数据库使用 MySQL 完成对数据的操作，包括增删改查，建立索引，建立存储过程等 [10]。

MQTT 服务器：移动端通过 MQTT 服务器与传感网对接，传感器节点发布（Publish）到指定主题（Topic），移动端订阅（Subscribe）对应主题（Topic）获取相应数据。使用 QoS Level 2 保证数据准确接收，并仅接收一次。

4.2 移动端设计

移动端使用 Visual Studio Code 作为开发环境，Dart 语言编写，Flutter 为跨平台框架，让船联网 APP 可以运行于 Android，iOS，Windows 等平台。

4.2.1 Dio 建立异步请求

在 Flutter 中，考虑到移动端有限运算能力，为提升用户体验，以保障 UI 流畅性为第一原则。因此，所有耗时操作，如连接网络，必须通过异步实现，本设计使用异步连接库 Dio 与服务端建立 Http 请求，实现登录功能，如图 4-2。

```
void _loginSync() async {
  try {
    Response response;
    var data = {'username': _usernameController.text,'password': _passwordController.text};
    response = await Dio().get("http://132.232.22.168:8080/ferry/users", queryParameters: data);
    if (response.data['status'] == "200") {
      if (response.data['data'] != null) {
        if (_loginButtonController.isCompleted) {
          setState(() {…
        } else {
          _loginButtonController.addStatusListener((status) {…
        }
        _loginSuccessController.addStatusListener((status) {…
        _loginSuccessController.forward();
        return;
      } else {
        showSnackBar(response.data['message'], 3);
      }
    } else {
      showSnackBar("未知错误", 3);
    }
```

图 4-2：使用 Dio 连接服务器

4.2.2 mqtt_client 连接 MQTT 服务器

在 Flutter 中，与 MQTT 服务器建立连接需要使用 mqtt_client 库，建立连接时需要设置 MQTT 服务器地址，端口，回调函数，保活时间，订阅主题，QoS Level，是否打印日志，是否清除缓存等，如图 4-3。

```
void connect() async {
  client = mqtt.MqttClient(mqttBroker, '');
  client.keepAlivePeriod = 30;
  final mqtt.MqttConnectMessage connMess = mqtt.MqttConnectMessage()
      .withClientIdentifier('Mqtt_MyClientUniqueId2')
      .startClean()
      .withWillQos(mqtt.MqttQos.atLeastOnce);
  client.connectionMessage = connMess;
  try {await client.connect();} catch (e) {_disconnect();}
  if (client.connectionState == mqtt.MqttConnectionState.connected) {
    connectionState = client.connectionState;
    debugPrint('MQTT已连接');
  } else {
    print('MQTT连接失败，状态为${client.connectionState}');
    _disconnect();
  }
}
```

图 4-3：使用 mqtt_client 连接 MQTT 服务器

4.2.3 高德地图绘制轨迹

本设计中显示位置与绘制轨迹功能需要使用地图功能，地图功能由高德地图提供支持。使用地图相关功能，需设置地图类型、绘制轨迹坐标位置、轨迹颜色、大地曲线等内容，如图 4-4。

```
//真实轨迹（红色实线）
var _realPostition = [originalPosition, position];
_controller.setPosition(target: position, zoom: 5);
_controller.addPolyline(
  PolylineOptions(
    latLngList: _realPostition,
    color: Colors.red[300],
    isGeodesic: true,
    lineCapType: PolylineOptions.LINE_CAP_TYPE_ROUND,
    width: 15,
  ),
);
originalPosition = position;
```

图 4-4：amap_base 设置地图属性和绘制航行轨迹

4.3 后端设计

船联网 APP 后台使用 IntelliJ IDEA 2019 进行开发，Java 语言编写。使用 Spring Boot 框架作为启动器，Gradle 为包管理工具，spring-mvc 为 MVC 控制器，持久层采用 Mybatis，数据库连接池为 Druid。

4.3.1 配置 Spring Boot

本系统使用 Spring Boot 作为框架，Gradle 为 Spring Boot 的依赖管理器。在 Spring Boot 中配置了 spring-mvc，mybatis，mysql-connector 的启动器，默认使用 Tomcat 作为动态服务器，如图 4-5 所示。

```
group = 'cn.luooyii'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '1.8'

repositories {
    maven { url 'http://maven.aliyun.com/nexus/content/groups/public/' }
    mavenCentral()
    maven { url 'https://repo.spring.io/milestone' }
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'com.fasterxml.jackson.module:jackson-module-kotlin'
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk8"
    implementation "org.jetbrains.kotlin:kotlin-reflect"
    implementation 'org.mybatis.spring.boot:mybatis-spring-boot-starter:2.0.1'
    runtimeOnly 'org.springframework.boot:spring-boot-devtools'
    runtimeOnly 'mysql:mysql-connector-java'
    compileOnly 'org.springframework.boot:spring-boot-configuration-processor'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
}
```

图 4-5：Gradle 中配置 Spring Boot 及 starter

4.3.2 RESTful API

为适应当下便捷式开发的理念，本系统采用 RESTful 的设计规范。RESTful API 相较于 SOAP 更为轻量级，具有松耦合，跨平台的特点。

系统完整请求流程如图 4-6 所示，移动端发送的请求经由 spring-mvc 拦截，会根据请求类型及请求地址调用 Controller 层对应方法，在 Controller 层调用 Service 层方法，对数据进行逻辑处理，最后响应请求，返回请求结果。

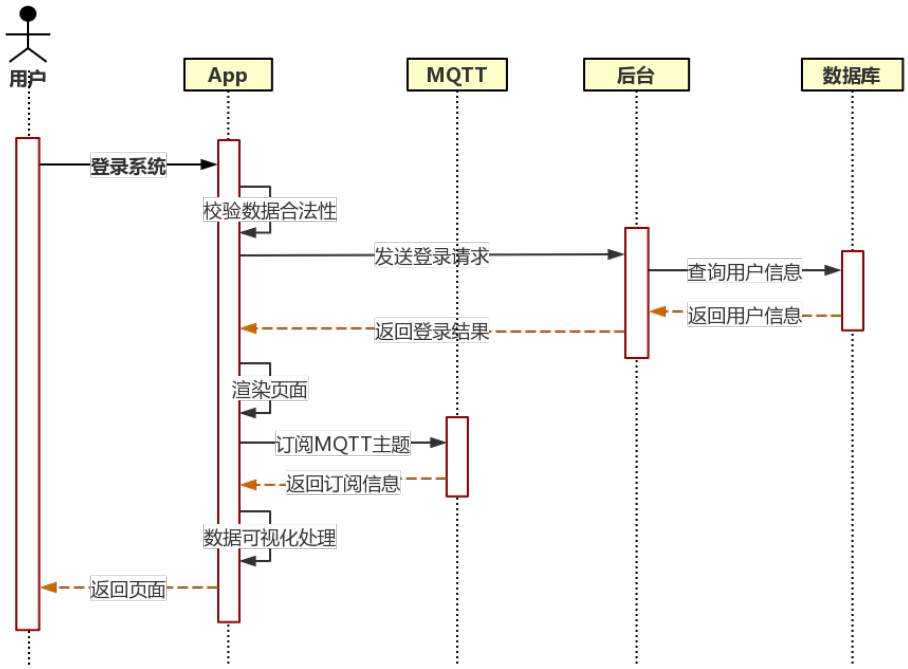


图 4-6：UML 时序图

4.4 数据库设计

数据库在整个系统中起到支撑作用，作为系统的底层服务和数据的核心，数据的设计、存储与维护，对系统的稳定性与可靠性至关重要。本系统采用 MySQL 数据库管理系统，MySQL 是当下应用最广的关系型数据库之一，其具有体积较小，成本低的特点。

本系统用户信息存储于数据库中，包括用户的登录名，密码，邮箱，类型等信息，在特定字段如用户名需添加唯一性约束，保障不会出现重名情况，如图 4-7 所示。

```
CREATE TABLE `sys_user` (
  `id` int(10) NOT NULL AUTO_INCREMENT COMMENT 'id',
  `name` varchar(50) DEFAULT NULL COMMENT '姓名',
  `login_name` varchar(50) DEFAULT NULL COMMENT '登录名',
  `password` varchar(225) DEFAULT NULL COMMENT '密码',
  `email` varchar(225) DEFAULT NULL COMMENT '邮箱',
  `usertype` int(1) NOT NULL DEFAULT '0' COMMENT '用户类型（0:用户1:管理员）',
  `create_time` varchar(225) DEFAULT NULL COMMENT '注册时间',
  PRIMARY KEY (`id`),
  UNIQUE KEY `login_name_UNIQUE` (`login_name`),
) ENGINE=InnoDB AUTO_INCREMENT=19 DEFAULT CHARSET=utf8 COMMENT='用户';
```

图 4-7：用户表创建语句

4.5 UI 设计

UI（User Interface 即用户界面）设计包括了软件设计学中的人机交互、操作逻辑、色彩、心理学等等内容。UI 是连接用户与系统的媒介，它实现了单纯的信息与人类感知之间的转换。优秀的 UI 设计不仅体现在界面的和谐，还需要改变软件的交互体验，让交互变得舒适、自由、体现出人文关怀与软件特点 [11]。

4.5.1 Material Design

本系统 UI 设计采用 Material Design 的设计理念，这是一种介于拟物化与扁平化之间的设计风格。Material Design 并不是纯粹的扁平化设计，它并没有抛弃掉阴影、纹理、速度等元素。或者说，它并不反对实体感。然而，Material Design 也不能简单归为拟物化设计。它的设计理念是在虚拟世界中带入物理世界的体验。筛去现实世界中实体的杂质和随机性，仅保留最原始形态、空间关系、过渡与变化，如图 4-8 所示，配合虚拟世界的灵活特性，还原最贴近真实的体验，达到简洁与直观的效果。

图 4-8：Material Design 设计风格

Material design 是最重视跨平台体验的一套设计语言。其设计规范严格细致而近乎严苛。Flutter 在技术上保证了跨平台的体验，而 Material design 用它严苛的设计规范保证了各个平台体验的高度一致。为了统一不同设备间的界面和交互，Material Design 没有如往常一样让像素处于同一个平面，而是按不同维度让它们规则处于空间当中。这样，保证了不同系统不同层面的元素，它们都是一致的、有原则、可预测的。高度的一致性与空间的和谐性不会让用户感到无所适从，也避免了开发者担心的视觉风格冲突。

4.5.2 色彩心理学

所有 UI 界面与人交互的因素中，色彩是最清晰、最强烈的，可以在第一时间传达设计者对事物的感受与情感的因素，它平等地、共通地沟通所有人。色彩的变化间会产生独特而微妙的情感变化，潜移默化中影响着人们的情绪。刘勰在《文心雕龙》中写到：“物色之动，心也摇焉。”他阐述了色彩对人心理活动的影响，色彩在刺激了人的视觉神经之后，会形成对颜色的感知，同时使大脑产生相应的情绪活动，这就是人对色彩的感情认识 [12]。

影响色彩的情感认知的一大因素是性别。多年来，众多 UI 设计师对颜色与性别之间的感情进行了深入研究，证明了男性与女性对色彩的情感认知不同，如图 4-9。在幼年时，男女通常会喜欢黄色，但随着年龄的撑场，他们对波短色彩（蓝色，绿色）比长波色彩（红色，橙色）更加喜欢。

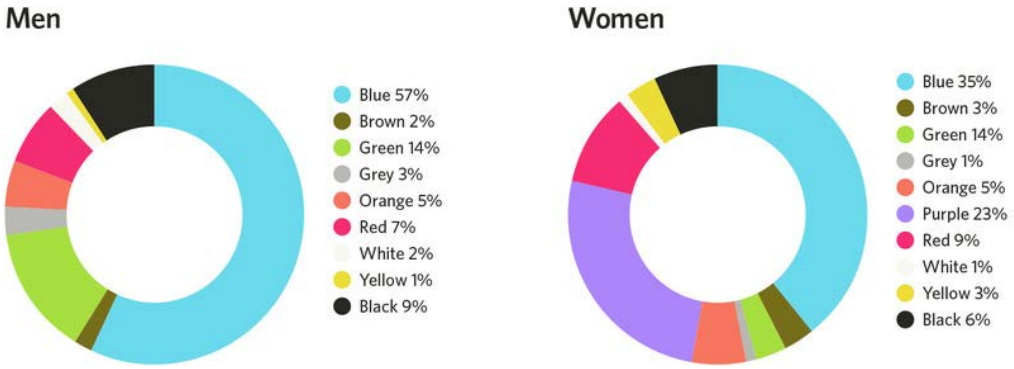


图 4-9：男性与女性最喜欢的颜色

本系统依据色彩心理学，综合考虑年龄、性别、情感认知等方面，选择了介于蓝色与绿色之间象征包容的海洋绿作为主色调，给人以深沉、平静的感觉，又辅以草绿色这种具有亲和力且易于识别的颜色，如图 4-10。这样的设计调节了 APP 的色彩平衡，给人以生机勃勃的感觉。配合船联网 APP 主要使用场景为宽阔的江面与一望无际的海洋，颜色的搭配更显和谐。清新而不乏深沉，互相包容，不会过分刺激大脑，给人以舒适耐看的感觉，使整个船联网 APP 看上去更为自然。

图 4-10：APP 色彩主题

第五章 系统实现

本章介绍船联网 APP 的系统实现相关内容，包括船联网 APP 的功能展示，使用 Docker 进行部署，腾讯云负载监控与 Flutter 性能测试等方面内容。展示了 APP 从源码到成品的完整过程。

5.1 移动端 APP

用户在打开船联网 APP——Ferry 后，进入到登录界面，如图 5-1(a)。用户根据船舶监管机构提供的账户信息登录船联网系统。在输入不存在的用户时，系统会给出提示“该用户不存在，请与平台联系解决。”当输入错误的密码时，给出提示“密码错误。”当输入正确的用户名和密码后，可进入船联网系统。

进入 APP 后可以看到 APP 两大功能模块——船舶监测模块与用户中心模块。用户可在位于底部导航栏右侧的用户中心模块，如图 5-1(b)，查看用户信息。或者在顶部工具栏打开用户菜单，如图 5-1(c)，查看简明用户信息。用户信息包括但不限于：用户昵称，邮件地址，用户头像等。

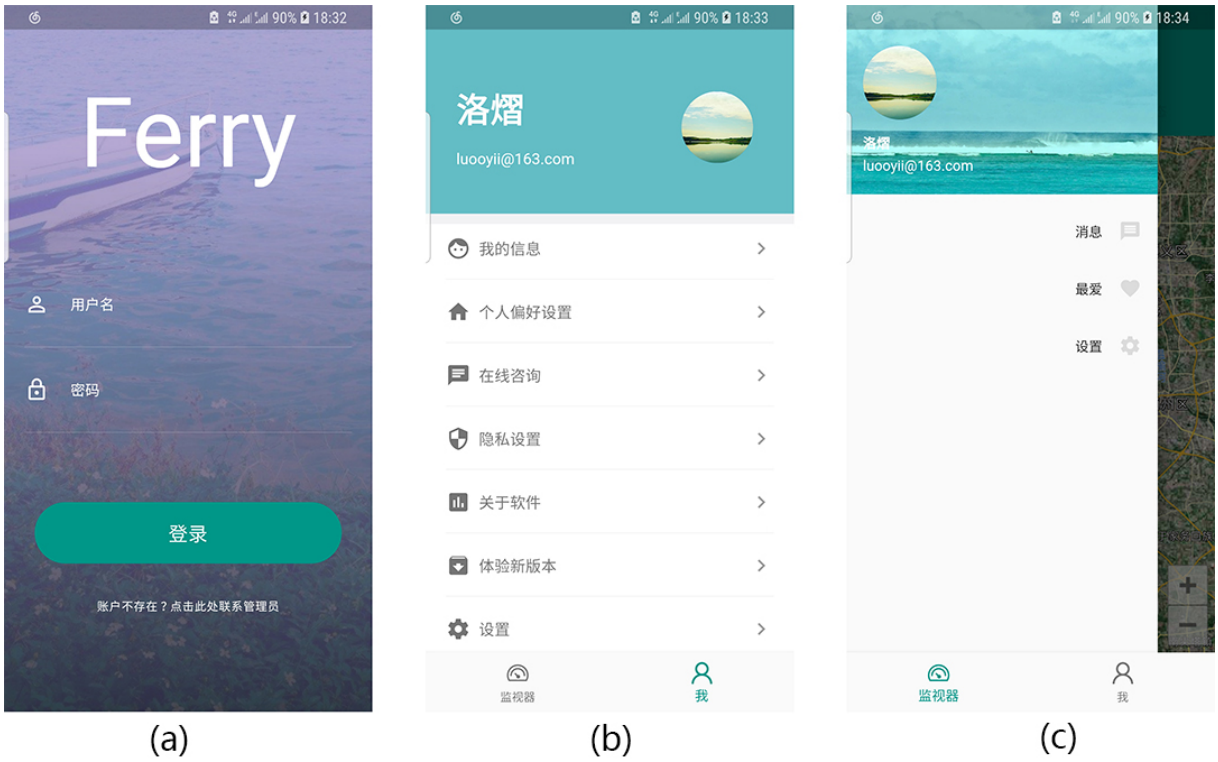


图 5-1: 登录、用户中心、用户菜单模块运行截图

船联网 APP 另一大功能模块为底部导航栏左侧的船舶监测模块，该模块下有三个子模块：电力监测、航行轨迹、船舶姿态，如图 5-2。

在电力模块中，APP 通过 MQTT 协议接收船舶上的电力数据，如电压、电流、功率等，如图 5-2(a)。接收到的电力数据会以折线图的方式动态显示，同时在折线图底部会显示具体数值。保障可以直观地了解船舶的电力状态，并在低电力时可以及时发现，及时处理。

航行轨迹模块，APP 通过 MQTT 获取船舶经纬度，并显示在由高德地图支持的地图控件上，如图 5-2(b)。当历史数据不断累积，地图控件可以绘制出船舶的航行轨迹。以实现精准定位和航线修正的功能。

船舶姿态模块，通过 MQTT 获取船舶姿态数据，可监测船舶的倾斜角度，如图 5-2(c)。APP 接收到船舶陀螺仪传来的 X 轴、Y 轴、Z 轴数据，进行计算，并使用 3D 建模功能对船体倾斜角度进行具象化的展示。提高用户与系统的交互性，并可以在船舶姿态失常时进行及时预警。

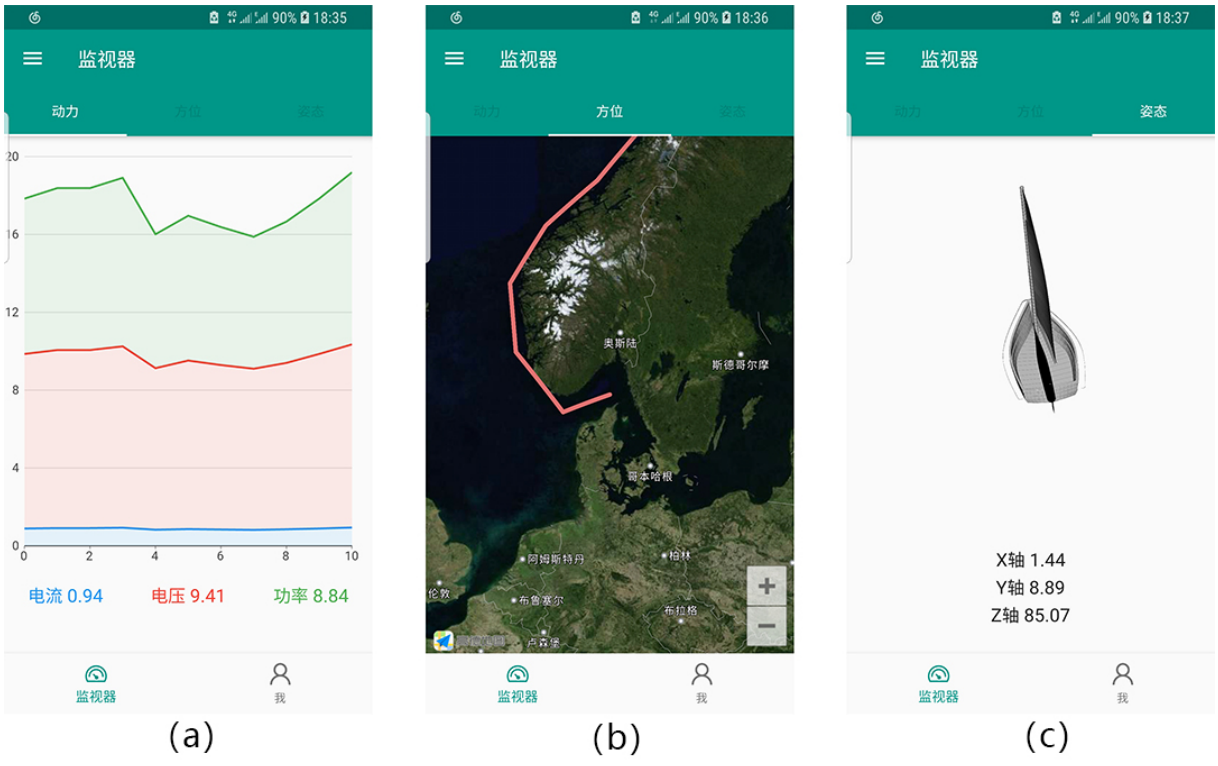


图 5-2: 船舶监控模块运行截图

5.2 Docker 部署

后台服务经 Gradle 打包后，生成 jar 文件，jar 文件与 Java 虚拟机镜像一同生成新的镜像。生成新镜像后，使用 docker run 创建一个新的容器运行镜像，同时指定容器与物理机的端口映射规则。容器成功运行效果如图 5-3 所示。

```
[root@VM_0_5_centos ~]# docker ps
CONTAINER ID        IMAGE                    COMMAND                  CREATED
a9cfad22b7a9        docker_zg_server         "java -jar ~/app.jar"    2 weeks ago
gnesi
3776129532c9        docker_ferry_server      "java -jar ~/app.jar"    3 weeks ago
d_roentgen
f5b8ac52314c        mysql                    "docker-entrypoint.s…"   5 weeks ago
ysql
72b196bfe1ff        postgres                 "docker-entrypoint.s…"   5 weeks ago
ostgre
dec702c9d886        nodered/node-red-docker  "npm start -- --user…"   2 months ago
```

图 5-3：查看已有镜像与正在运行的容器

5.3 负载监控

系统使用腾讯云平台监测系统运行状况（CPU 负载、内存使用量、外包数量与磁盘 IOPS 等）如图 5-4 所示。良好的监控保障系统的稳定运行，避免了可能出现的网络灾难。

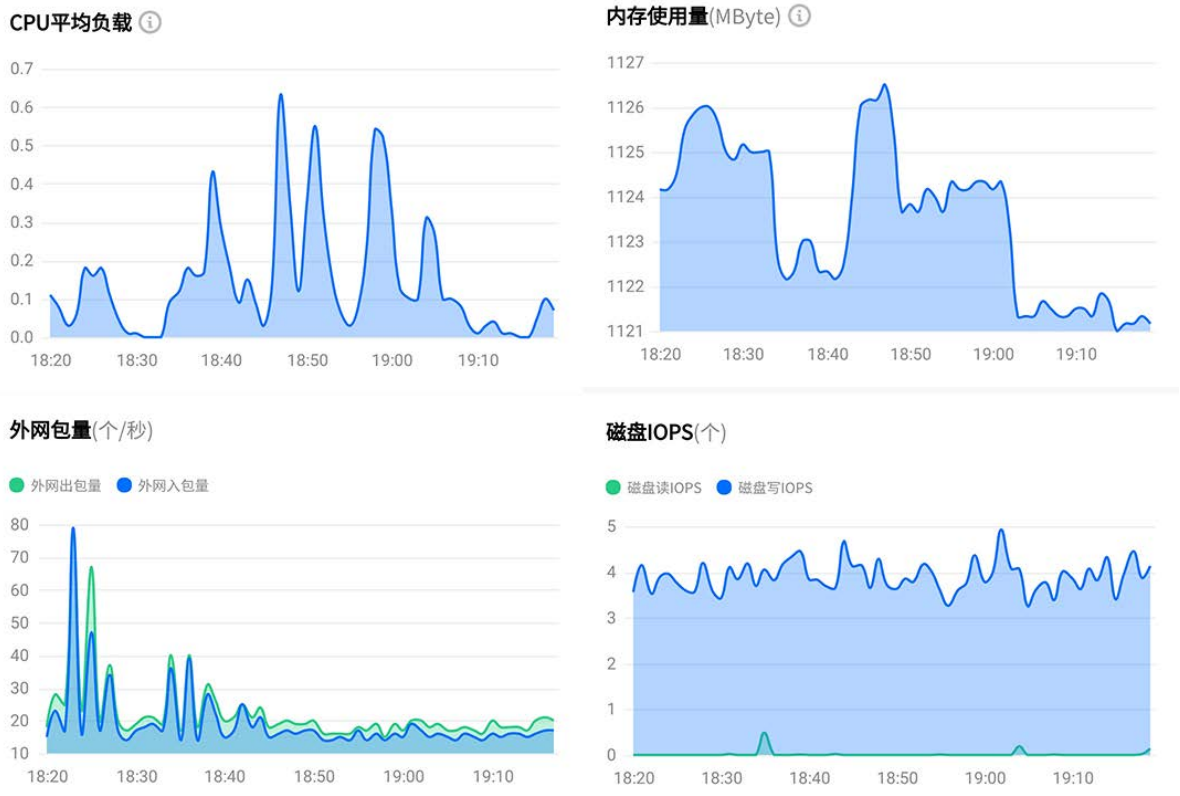


图 5-4：服务器负载状况

5.4 性能测试

本设计采用 Flutter 作为移动端框架。在性能监测方面，Flutter 有专门为监控 APP 性能开发的 Profile 模式。在 Profile 模式下，一些扩展的服务功能（Service extensions）是打开的，例如监控性能的浮层等。在此状态下可以实时监测 APP 的内存状况如图 5-5(a)和 CPU/GPU 状况如图 5-5(b)。

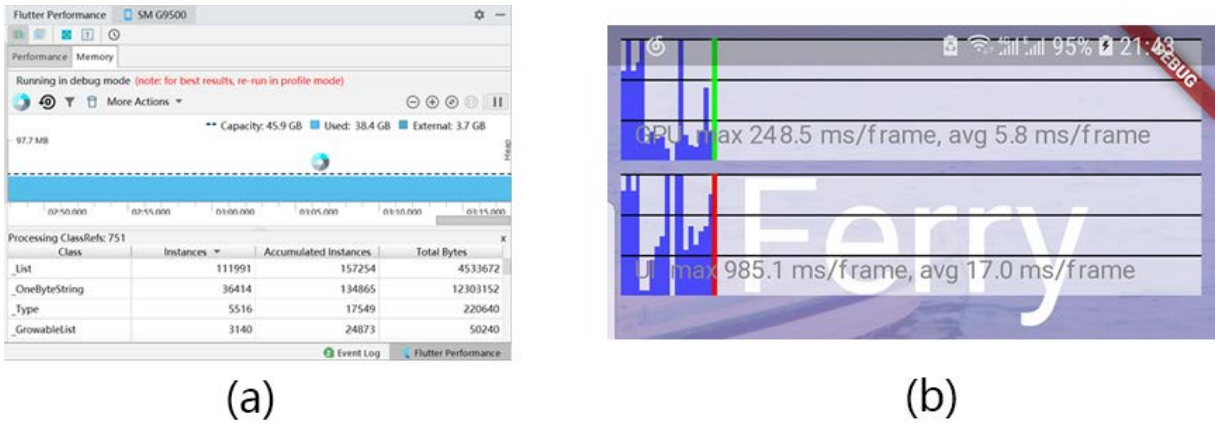


图 5-5: 内存、帧率、CPU、GPU 运行状况

通过实时性能监测，可以发现 Flutter 在内存、CPU 资源占用方面有媲美原生的性能。优秀的性能控制保障了船联网 APP 在低性能设备上也具有高可用性，使船联网 APP 的使用范围更加广泛。同时更优秀的性能表示着船联网 APP 在使用时的启动速度与连接时延更短，响应更迅速，可以给用户带来良好的使用体验。

结 论

船联网的出现将会变革当下的船舶信息服务领域，极大的提升船舶行业的用户体验。本论文主聚焦于船联网 APP 的研究设计，分析的 APP 与船联网系统、与用户之间的交互，旨在提高船联网给与用户的体验。在此过程中本设计主要做出以下工作：

首先，APP 对纯粹的数据进行可视化处理，提高数据可读性。使用户可以更加直观地了解船联网系统的运作，拉近了用户与系统的距离。其次，考虑到当下多样的终端设备，对于跨平台做出了特别考虑。用户可以在任何地方以任何终端与船联网系统进行交互，协同进行工作。同时，考虑到在航行时复杂的网络环境，APP 选择利用轻量、开源、易于实现的 MQTT 协议保障数据传输。

最后，APP 重新设计了 UI 交互，色彩是最清晰、最强烈的，可以在传达情感的因素。本设计根据色彩心理学，选择了象征包容与平静的海洋绿作为主色。在辽阔的江面与一望无际的海洋上，更能安抚使用者的内心，贴近于用户的距离。

本文对船联网 APP 的可视化，交互，安全性，可靠性方面进行深入的研究，有效提升了船联网系统的使用体验。但是受工程实践、技术、时间等因素的影响，本设计仍有可进一步完善的空间。因而在此提出以下展望：

未来，大量的数据将会在船联网平台上产生，数据的本身具有巨大的价值，结合机器学习与云计算，预测船舶状态，智能控制电力。以此创造出巨大的经济效益。随着加入船联网平台的船只与航线越来越多，船联网可根据航线状况对船舶进行分流与控制，保障航线的通畅，提高航运效率。最后实现船舶实时信息共享，对海洋资源，突发状况可以更加迅速有效地处理。

通信技术在不断发展，各种新的事物不断出现，船联网行业也会不断注入新的活力。社会同样期待着，更加智能、更加人性化的船联网系统的出现，来推动航运业的不断发展。

致 谢

随着毕业论文的结束，我的大学生活也逐渐进入了尾声。当写下这段文字，经历了社会磋磨后的我更加珍惜最后的学生时光。生活总是能逼迫着我不断成长，而我也仅能感谢生活予以我的一切。回顾往昔，在迷茫与困惑中，幸而能够有所坚持。仅借此毕业论文完成之际，向一路上予以我帮助的老师、同学、家人、路人们致以诚挚的感谢。

首先感谢在我论文完成的过程中给予我指导的张笑非老师与房靖老师。感谢张老师对我毕业设计提出的新的思路，感谢房靖老师对本设计传输过程中安全性的建议。在完成毕业设计的过程中，我学到了许多以前没有学到的知识，这些知识对我的实习产生了很大的帮助，让我可以迅速适应工作的需求。这次船联网 APP 设计是我第一次独自进行如此全面和系统的设计工作，过程中不免有疏漏之处，感谢张老师为我指正了许多言辞中不曾留意的细节。

感谢无数默默奉献的开源社区贡献者们，你们倾注的热情、时间、智慧才得以让软件行业发展至今。前人探索而后人传承，此谓开源之精神所在。本系统相关成果将在毕业设计完成后上传至 GitHub，仅以此致敬无数默默为开源界和软件发展付出的贡献者们。GitHub Repositories : https://github.com/luooyii/ferry.

时光荏苒，感谢多年学习生涯带给我的一切，有幸接触到了古今中外经年累月传承于今的种种思潮。让我深感自身的渺小，也敦促着我不断思考。愿此论文将不会是我学术思考的终点。

最后，感谢百忙之中参加评审的老师们，对这份还极不完善的设计的报以宽容，并对此提出了宝贵的建议。

参 考 文 献

[1] 彭友. 基于船联网的智能船舶协同综合保障平台的设计与实现[D].江苏科技大学,2017.

[2] 程子清.Flutter 开发智慧城市相关 APP 具体实践[J].电子技术与软件工程,2018(19):56-57.

[3] 周勇,程子清.Flutter 的原理深度剖析[J].电脑编程技巧与维护,2018(11):19-21.

[4] 刘劭. 基于微服务的教学支持平台服务端的设计与实现[D].南京大学,2018.

[5] 马雄. 基于微服务架构的系统设计与开发[D].南京邮电大学,2017.

[6] 袁青杰. 智能家居系统扩展及应用服务的技术研究[D].西安电子科技大学,2018.

[7] 白昊. 基于 MQTT 协议的物联网平台设计与实现[D].西安邮电大学,2018.

[8] 金诚.移动应用跨平台开发框架的比较分析[J].民营科技,2018(10):150-152.

[9] 王玉洋. 基于微信小程序的移动学习平台环境构建与系统设计开发[D].南京大学,2018.

[10] 汤汪艳. 基于用户体验的手工艺微信小程序设计研究[D].中国美术学院,2018.

[11] 杨沉. 视觉符号在手机界面设计的应用研究[D].江西师范大学,2016.

[12] 何浩.基于色彩心理学的 UI 设计研究[J].中国新通信,2015,17(09):11.