第五周进度报告

一、代码实现讨论

（1）本周迭代：对原设计的推翻与继承

- 继承：继续坚持“服务端（Java Netty + C++ JNI 业务）/客户端（Qt）”的C/S分层；数据库仍使用MySQL，JSON为统一交换格式；命令式TCP协议保持行分隔（\n），不改变既有协议栈。

- 推翻与重构：

- Qt 管理员界面 AdminTab 的数据解析逻辑重构，统一按后端最新JSON结构（如 data.user\_coupons、data.low\_stock\_products.products、data.promotions）进行解析，避免字段漂移导致的空表。

- 发券流程端到端梳理：AdminTab 发起 ASSIGN\_COUPON → Netty 解析 → JNI C++ CouponService.assignCouponToUser 校验（支持“满减50元券 (SAVE50)”等复合输入，自动提取 code）→ DataBase 落表 → 返回规范JSON；客户端收到后即时刷新“可发放列表”和“用户券”。

- 强化健壮性：所有 AdminTab 的 sendCommand 回调统一做 JsonUtils::isSuccess 检查与 message 提示，避免异常路径静默失败。

- 增加：在发券流程中，进一步优化了 JSON 数据的校验逻辑，确保字段缺失时能够提供默认值，避免客户端崩溃。

- 增加：对 AdminTab 的界面交互进行了微调，提升了用户体验，例如在加载数据时增加了进度指示器。

（2）MVC/分层实践体验

- 服务器端：

- 表现层：Netty 命令处理器（会话、鉴权、协议解析）。

- 业务层：JNI 暴露的 C++ 服务（UserService/ProductService/CartService/CouponService/OrderService 等），聚合在 EmshopServiceManager 单例中。

- 持久层：C++ 中统一的 SQL 执行与结果封装（executeQuery/executeUpdate），由连接池 DatabaseConnectionPool 支持。

- 客户端：

- View：Qt Widgets（MainWindow、Dashboard/Products/Cart/Orders/Admin 各 Tab）。

- Controller：各 Tab 里的槽函数（按钮/选择/定时器事件）+ NetworkClient 负责协议收发。

- Model：轻量以 JSON 映射到 UI 控件（未引入复杂的 Qt Model/View 数据模型以控制复杂度）。

- 体验：职责边界清晰，迭代时仅在 AdminTab 与服务端JSON契约处修改；其他模块零改动，验证分层带来的可维护性与回归成本可控。

- 增加：在客户端的 Controller 层中，进一步细化了事件处理逻辑，将复杂的业务逻辑下沉到独立的工具类中，提升了代码的可读性和复用性。

- 增加：在服务端的持久层中，为 executeQuery 方法增加了超时机制，避免长时间阻塞。

（3）设计模式使用

- 单例：EmshopServiceManager、DatabaseConfig、DatabaseConnectionPool；统一资源与配置管理。

- 工厂/注册表：服务管理器内部按需创建服务实例。

- RAII：数据库连接与锁的自动释放，避免泄漏。

- 模板方法：BaseService 提供通用查询/错误包装骨架，各具体 Service 复用。

- 策略/组合：促销系统预留多策略（折扣、满减、买N送M、免运费等），在 Java 侧以类型字段 + conditions 驱动，客户端透明展示。

- 观察者（信号槽）：Qt 信号槽贯穿 UI 事件与网络回调，弱耦合。

- 增加：在促销系统中，进一步完善了策略模式的实现，新增了“限时折扣”策略，并通过配置文件动态加载。

- 增加：在客户端的信号槽机制中，增加了对异常情况的捕获和日志记录，确保问题可追溯。

（4）多线程使用

- 服务器：

- Netty 多线程 I/O 事件循环；

- C++ 连接池含维护线程（定期健康检查与回收），并通过互斥锁与原子变量保障并发安全；

- 业务执行依赖连接池并发复用，避免阻塞。

- 客户端：

- Qt 网络在事件循环线程内处理，UI 线程与网络 I/O 线程通过信号槽通信；

- 低库存自动刷新使用 QTimer，避免阻塞 UI。

- 增加：在服务器的连接池维护线程中，新增了动态调整连接数的功能，根据当前负载自动扩展或收缩连接池大小。

- 增加：在客户端的网络模块中，优化了信号槽的线程切换逻辑，减少了不必要的线程上下文切换，提高了性能。

（5）经典处理技术

- 端到端协议容错：

- 客户端解析容忍多版本字段名，例如 status/user\_coupon\_status、end\_time/expire\_at、data.promotions 或 data 数组等；

- 服务器对“名称+代码”的复合输入进行鲁棒解析（"满减50元券 (SAVE50)" → 提取 code=SAVE50），提升可用性。

- 数据库连接池：

- 初始化多连接、超时/重连、定期清理与健康检查，记录连接生命周期日志，避免高并发下雪崩。

- 统一 JSON 响应协议：success/message/data/timestamp 四段式，便于前后端调试与人类可读日志。

二、本周技术问题与解决

- 问题1：发放优惠券后客户端“无券显示/崩溃”

- 现象：添加或分配优惠券后，AdminTab 券列表不刷新或程序异常退出；Windows 事件查看器出现 0xc0000374（堆损坏）记录。

- 分析：客户端解析字段与后端返回不一致，导致空对象访问；另外命令行一次性启动/切换目录出错，引发误判；未复现到真实的内存越界代码，Qt 侧未见危险的裸内存操作。

- 解决：

1) 统一 AdminTab JSON 解析路径与兼容分支；

2) 发券成功后强制刷新“可发放列表+用户券”；

3) sendCommand 增加失败提示，避免静默；

4) 运行建议改用 VS Code 任务“Run Qt Client”，确保 PATH/工作目录一致，降低环境因素导致的异常。

- 问题2：库存页“阈值10仍显示非低库存商品”

- 分析：后端返回 data.low\_stock\_products.products 是全量列表并带 is\_low\_stock 标志；客户端单用阈值判断会误标。

- 解决：优先用 is\_low\_stock；若字段缺失，再回退用 stock<=threshold。

- 问题3：促销活动列表不显示

- 分析：服务端返回 data.promotions；客户端旧逻辑读取 data。

- 解决：先读 data.promotions，回退 data；展示 details 使用紧凑JSON。

- 增加：

- 问题4：客户端在高并发情况下出现 UI 卡顿现象。

- 分析：由于主线程处理了过多的网络回调，导致界面刷新不及时。

- 解决：将耗时的网络回调逻辑移至后台线程，并通过信号槽机制通知主线程更新 UI。

- 问题5：服务端在处理大批量数据时出现内存溢出。

- 分析：由于未对查询结果进行分页处理，导致内存占用过高。

- 解决：为所有大数据查询增加分页支持，并限制单次查询的最大记录数。

三、后续优化想法

- 客户端

- 引入 Qt 的 Model/View（QAbstractTableModel）重构表格，减少 setItem 频繁创建的开销；

- 增加非阻塞状态展示（状态栏+Toast），并对失败分支提供可见提示；

- 为管理操作增加确认对话框与结果弹窗，减少误操作。

- 增加分页功能，避免一次性加载过多数据导致的性能问题。

- 增加用户操作日志记录功能，便于问题追踪和审计。

- 服务端

- 协议统一JSON化：为命令行协议增加 action 字段，逐步与行文本命令并存，便于扩展与参数复杂化（CREATE\_PROMOTION等）；

- 增加速率限制，防止恶意请求导致的服务不可用。

- 优化日志记录格式，增加唯一请求 ID，便于关联分析。

- 数据层

- 针对高频查询（商品/券）增加只读从库或本地缓存（LRU），并结合失效策略；

- 引入 Redis 缓存，减少高频查询对数据库的压力。

- 为关键表增加索引优化查询性能。

四、下周计划与方案

- 计划1：完成“促销创建”命令全链路

- 任务：在 Netty 增加 CREATE\_PROMOTION 命令分派；JNI/C++ 实现 createPromotion(json) 写库与校验；Qt Admin 新增创建后刷新与表单校验。

- 验证：新增/冲突码/必填缺失用例，回归 GET\_ACTIVE\_PROMOTIONS 展示。

- 计划2：订单详情/状态流编辑对话框

- 任务：Qt 端为订单“改状态/详情/退款”接入弹窗与确认流程；服务端补充 VIEW\_ORDER 更详尽文本/JSON。

- 验证：多状态流转与权限校验（仅管理员）。

- 计划3：日志与稳定性

- 任务：客户端加 qInstallMessageHandler 定向到文件；服务端细化异常捕获并将 SocketException（Connection reset）视为正常断开。

- 验证：异常路径日志可检索；长连压测下稳定退出。

- 计划4：优化客户端的错误提示机制。

- 任务：为所有可能的错误场景增加用户友好的提示信息，并提供重试选项。

- 验证：模拟各种错误场景，确保提示信息准确且易于理解。

- 计划5：提升服务端的高并发处理能力。

- 任务：为 Netty 的线程池增加动态调整功能，根据当前负载自动扩展或收缩线程数。

- 验证：通过压测工具模拟高并发场景，确保服务稳定性。

更详细项目信息：[AshIgnis/JLU\_Emshop\_System](https://github.com/AshIgnis/JLU_Emshop_System)

55240425

屈熙宸