合肥工業大學

计算机与信息学院

《软件工程师综合训练》报告

设计题目：某送水公司的送水系统

项目组长：余梓俊

成　　员：姜润泽、秦鹏飞

专业班级：计算机科学与技术 18-3 班

指导老师：胡敏

承担任务：负责代码框架和环境搭建、前端实现
参与需求分析、数据库设计、后端实现

2021 年 10 月

目 录

一、系统开发概述 1
1.1 项目背景 1
1.2 编写目的与实现意义 1
二、系统规划 2
2.1 设计任务要求 2
2.2 软件环境与工具 2
三、需求分析 3
3.1 用户需求说明 3
3.1.1 数据需求 3
3.1.2 事务需求 4
3.2 系统需求说明 4
3.2.1 系统特性 4
3.2.2 系统总体结构图 5
3.2.3 数据流图 5
四、可行性分析 7
4.1 经济可行性 7
4.2 技术可行性 7
4.2.1 后端接口风格 7
4.2.2 前端技术选型 7
4.2.3 后端技术选型 8
四、数据库概念和逻辑设计 9
4.1 ER 图 9
4.2 数据字典 12
4.3 关系表 15
五、数据库物理设计 17
5.1 触发器 17
5.2 存储过程 17
5.3 视图 18
5.4 数据完整性 19
5.4.1 DEFAULT 约束 19
5.4.2 CHECK 约束 19
5.4.3 外键约束 19
5.4.4 其他机制 19

六、应用程序设计 21
6.1 功能模块 21
6.1.1 登录模块 21
6.1.2 信息管理模块 23
6.1.3 费用管理与统计模块 23
6.2 界面设计 25
6.2.1 总体设计 25
6.2.2 登录模块 26
6.2.3 信息管理模块 26
6.2.4 费用管理与统计模块 27
七、总结 29
附. 参考文献 30

一、系统开发概述

送水管理系统是一个面向社会桶装水销售点的信息管理平台，该系统集合各种管理功能于一体，从而提高了桶装水销售的效率，为管理者对数据管理提供方便，同时对数据进行分析，以便调整销售策略。

1.1 项目背景

随着人们生活水平的提高，桶装水已经成为人们生活中的必需品，企事业单位、学校、银行、医院、家庭等等各类用户都使用桶装水，水是人们生活中必不可少的，因此近年来桶装水的消费数量迅速增长，这为桶装水行业带来了很好的发展机遇，也预示着这个行业广阔的发展前景。同时，也要求了桶装水配送行业不断的提升自我，提高工作效率，能够快速、有序的运作。而现有桶装水店大部分是多品牌、多品类经营，管理头绪多、漏洞多、管理复杂，因此科学的管理成为桶装水配送这个行业的关键。因此一个好的送水管理系统应势在必行。

1.2 编写目的与实现意义

和组员一起作为一个团队设计一个完整的软件工程项目，是对我们所学软件知识的一次大综合。设计送水管理系统，可以帮助我们巩固复习软件工程与数据库系统两门课中所学到的知识；实现送水管理系统，可以大大提高我们实践能力与分析解决实际问题的能力；在此过程中，我们必将遇到从未见过的问题，用到未曾学习过的知识和技能，实现这样一个系统也将提高我们的自学能力。通过实践，我们能更深刻地体会软件工程方法学的具体应用，了解软件项目的方方面面。

二、系统规划

2.1 设计任务要求

基本功能要求：

(1) 实现工作人员、客户信息的管理；

(2) 实现矿泉水类别和供应商的管理；

(3) 实现矿泉水入库管理和出库管理；

(4) 实现费用管理；

(5) 创建触发器，实现入库、出库时相应类型矿泉水的数量的增加或减少；

(6) 创建存储过程统计每个送水员工指定月份送水的数量；

(7) 创建存储过程查询指定月份用水量最大的前 10 个用户，并按用水量递减排列；

(8) 具有数据备份和数据恢复功能。

2.2 软件环境与工具

开发工具：

平台：Ubuntu 20.04 (on WSL2)

IDE：VSCode

团队协作与版本控制：Git

Linter: ESLint (Airbnb Style)

前端：

语言：JavaScript (ES6)

框架：Vue2

组件库：element-ui

后端：

语言：Node.js

框架：Express

数据库：MySQL

三、需求分析

现今生活中人们对桶装饮用水的需求量很大，怎样有效、快捷、有序的进行桶装水的配送成为一个桶装水公司成功的关键，而手工管理不能满足桶装水企业快速发展的需求，因此对于桶装水公司来说一套面向社会桶装水销售点的送水管理软件是必须的，该软件需集合各种管理功能为一体，提高桶装水的销售效率，减少不必要的劳动，为桶装水公司节省开支。

3.1 用户需求说明

系统的使用者为送水公司的管理员，管理员登录系统，可以查看各种类型的矿泉水的库存，可以进行送水工、客户信息、供应商信息的管理，录入相应的进货单和出货单，查看进货销售的各种统计信息。

3.1.1 数据需求

经过我与组员们的讨论，以及指导老师在验收时的补充，我们认为，系统使用者，即送水公司的管理员，至少有如下的数据需求：

- 对于送水工的管理，需要工人的工号，姓名，性别，年龄，入职时间，电话号码；
- 对于不同类型的矿泉水，需要矿泉水的类型号，该类型的描述信息，该种水的计量单位，如桶、瓶、升（如果计量单位为桶或瓶等，需要记录一单位的矿泉水的具体量），该种矿泉水当前的库存；
- 对于顾客，需要记录顾客的姓名，电话，住址，并允许添加备注信息；
- 对于供应商，需要记录其名字，地址，并允许添加备注信息；
- 对于进货单，从一家供应商进一次货应当产生一张进货单，管理员录入进货单进系统，进货单应有进货单号，日期，供应商，一张进货单里可以有若干进货条目，每条进货条目应当写明进了何种矿泉水，计量单位是什么，一单位的量为多少，进了多少单位，以及单位价格，最后，一张进货单应当写明该进货单的总价格；
- 对于出货单，与进货单类似，一位顾客下单一次应当产生一张出货单，某位送水工将负责该单的配送，管理员录入出货单进系统，出货单应有出货单号，日期，顾客，送水工，一张出货单里面可以有若干出货条目，每条出货条目应当写明订

购了何种矿泉水，计量单位是什么，一单位的量为多少，订购了多少单位，以及单位价格，最后，一张出货单应当写明该出货单的总销售额。

3.1.2 事务需求

我们认为，该系统的使用者，即送水公司的管理员，至少有如下的事务需求：

- 管理员应使用密码登录系统，并且使用账号密码是进入系统的唯一方式，不能通过例如输入后台管理的网址来绕过登录界面，系统不能有安全漏洞；
- 录入入库单和出库单时，管理员录入完该单的所有条目后，系统应当自动计算出该单的总金额，而不是由管理员手动输入；
- 录入入库单或出库单后，各种类别的矿泉水的库存应当自动增加或减少；
- 录入一个完整的入库单或出库单，应当是一个原子性的操作，如果录入入库单或出库单中的某一条目时出现问题，该单的所有条目都应该不被录入系统，并向管理员提示再次完整地录入该单；
- 系统应当能产生进货和销售的各种统计信息，例如，某段时间的进货开销、卖水收入，送水工在某段时间的送水量，顾客在某段时间的订水量，等等。

3.2 系统需求说明

该送水系统，能够借助计算机实现半自动化的管理，方便送水公司的管理员进行出入库和财务管理，能够帮助送水公司简化管理流程，减少录入和统计过程中可能发现的错误，减少人力物力成本。要求系统具备以下特点：

(1) 操作简单，易用。

(2) 数据存储可靠，具备较高的处理效率。

(3) 系统安全、稳定。

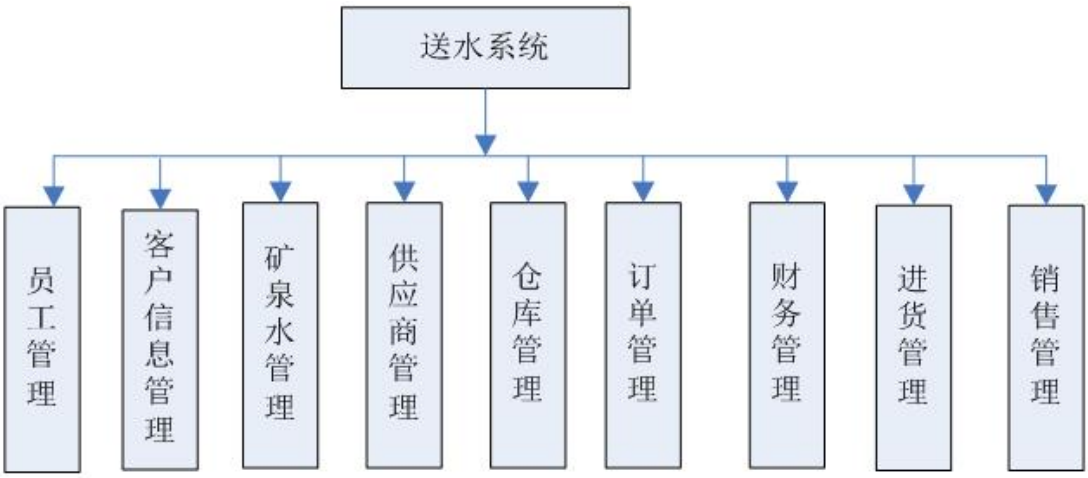
(4) 开发技术先进、功能完备、扩展性强。

3.2.1 系统特性

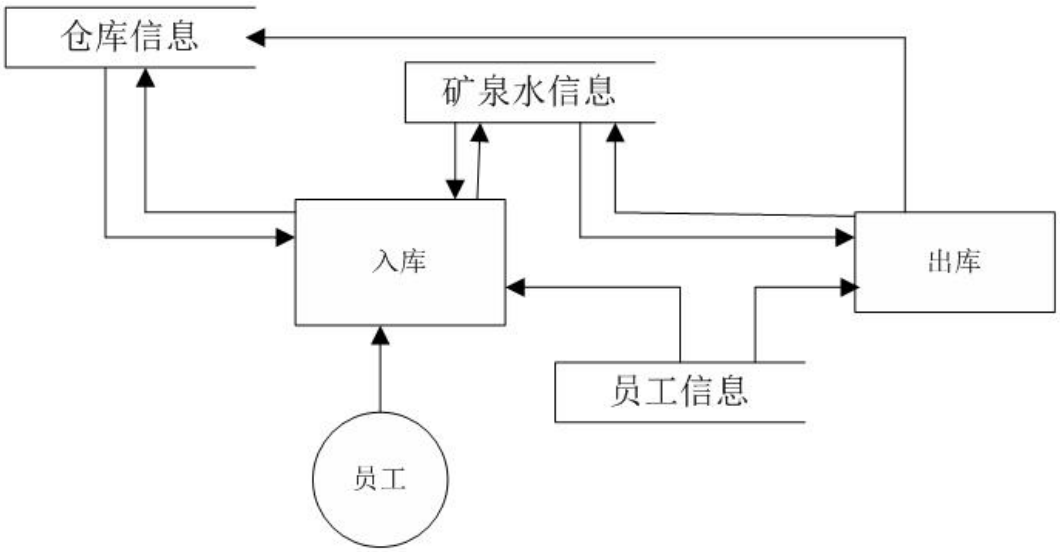
- 可移植性：系统可以在各种主流的平台上运行，对于前台界面程序，要能在常见的用户终端上使用，如 Windows 平台、Mac 平台等，对于后台服务，要能部署在流行的服务器架构上并稳定运行，如 Windows Server，Ubuntu，Cent OS 等。

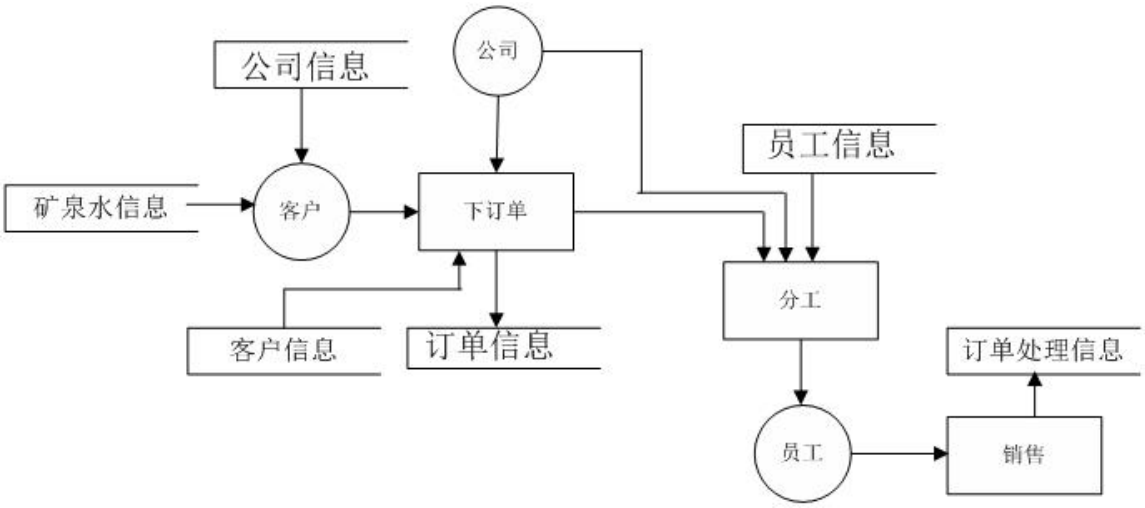
- 安全性：使用账号密码是进入管理系统的唯一方式，任何非法的访问应当被拒绝；所有不当的操作应当被拒绝执行，错误的数据应当被拒绝写入。
- 可用性：录入订单是原子性操作，不会出现订单录入不完整的情况；一旦系统出现故障，非硬件故障可以立即重启系统，持久化的数据不会丢失。
- 灵活性：从底层的数据表示的设计，后台接口的设计，前台的布局设计，三方面考虑未来业务需求的增长和变化，提供良好的可修改性和扩展性。

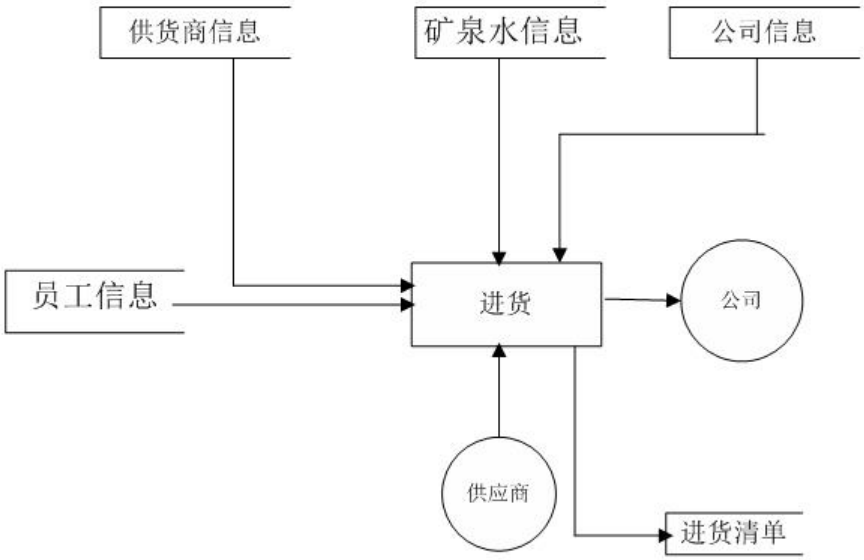
3.2.2 系统总体结构图



3.2.3 数据流图




公司信息
公司
员工信息
矿泉水信息
客户
下订单
分工
客户信息
订单信息
订单处理信息
员工
销售


供货商信息
矿泉水信息
公司信息
员工信息
进货
公司
供应商
进货清单

四、可行性分析

送水管理系统使桶装水销售网点以及桶装水公司简化了管理流程，提高了桶装水销售的效率，同时为管理者对数据管理提供方便，下面从经济可行性以及技术可行性对送水管理系统进行详细的论证。

4.1 经济可行性

目前各桶装水公司以及桶装水销售网点采用人员手工管理，而现今生活中对桶装饮用水的需求量很大，而员工每天处理的数据量是有限的，这就给公司以及网点增加了人工成本，同时人难免会出现疏漏和错误，可能带来不可估计的损失。

送水管理系统解决了桶装水公司和送水网点所面临的窘境，简化了人员操作，减少了人工成本，提高了效率，同时便于管理者进行管理，减少了人员的疏漏带来的损失。送水管理软件具有很强的实用性，以及广阔的市场前景。因此该送水管理软件在经济上是可行的。

4.2 技术可行性

本系统采用 B/S 架构，前后端分离，结构清晰，开发效率高，方便团队合作。

4.2.1 后端接口风格

在我们的前后端交互中，我们采用了 RESTful 架构。REST 是 Representational State Transfer 的缩写,如果一个架构符合 REST 原则,就称它为 RESTful 架构。RESTful 架构可以充分的利用 HTTP 协议的各种功能，是 HTTP 协议的最佳实践；RESTful API 是一种软件架构风格、设计风格，可以让软件更加清晰，更简洁，更有层次，可维护性更好。

4.2.2 前端技术选型

我们的前端框架采用了流行的 Vue.js。Vue 是一套用于构建用户界面的渐进式框架。与其它大型框架不同的是，Vue 被设计为可以自底向上逐层应用。Vue 的核心库只关注

视图层，不仅易于上手，还便于与第三方库或既有项目整合。另一方面，当与现代化的工具链以及各种支持类库结合使用时，Vue 也完全能够为复杂的单页应用提供驱动。

4.2.3 后端技术选型

我们的后端语言选择了 Node.js，作为一个动态类型的脚本语言，它可以使我们快速地开发出可用的后台服务，减少在各种配置上花费的时间以及在编译过程中可能出现的各种问题。只要机器上安装了 Node.js 运行时环境，只需要使用一条命令就可以启动服务进程，方便我们专注于开发和调试。

在后端框架上，我们选择了 Node.js 平台的经典框架 Express。Express 是一个保持最小规模的灵活的 Node.js Web 应用程序开发框架，提供一系列强大特性帮助开发者创建各种 Web 应用。Express 不对 node.js 已有的特性进行二次抽象，只是在它之上扩展了 Web 应用所需的功能，它拥有丰富的 HTTP 工具以及来自 Connect 框架的中间件随取随用，可以让我们简单快速地创建 API。

在数据库管理系统上，我们选择了 MySQL。在 WEB 应用方面，MySQL 是最流行的、最好的关系型数据库管理系统之一。MySQL 有大量的可参考资料，体积小、速度快，并且有一个开源的免费的社区版本。各大云服务商也均提供 MySQL 的云数据库，大型的国产数据库也都兼容 MySQL 语法，例如阿里云的新一代关系型数据库 PolarDB。因此选择 MySQL 可以方便我们后期将数据库上云，改用国产数据库，等等。

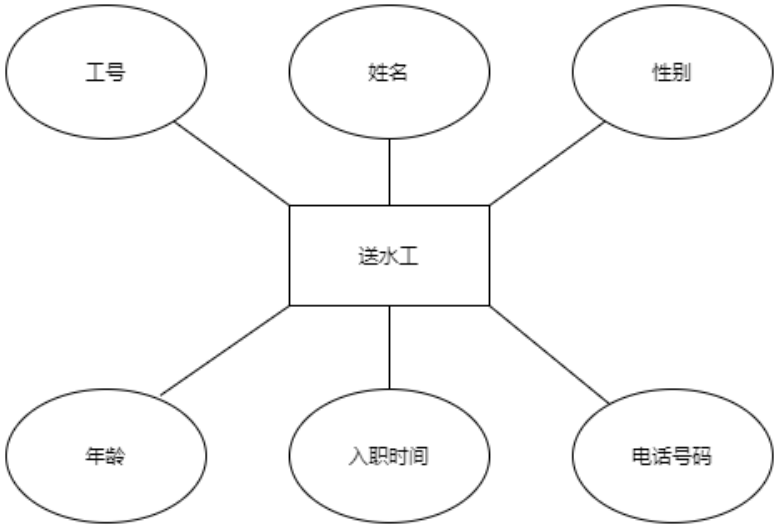
四、数据库概念和逻辑设计

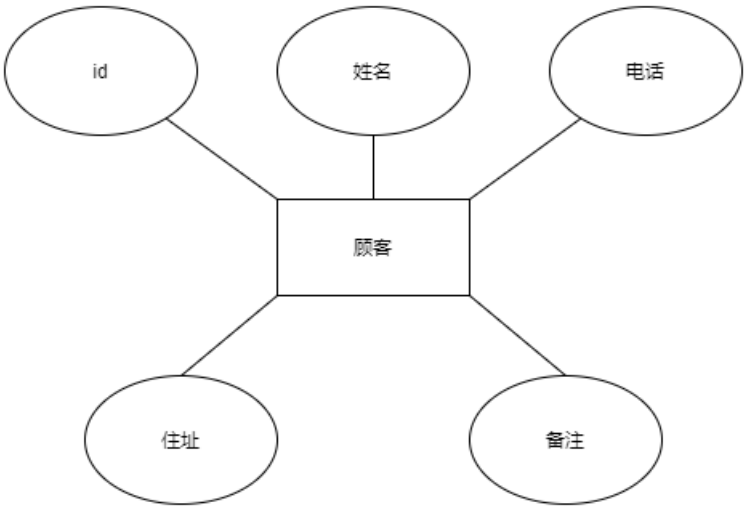
```
Database changed
mysql> show tables;
+--------------------+
| Tables_in_songshui |
+--------------------+
| Customers          |
| PurchaseOrderItems |
| PurchaseOrders     |
| Purchase_Daily     |
| SellOrderItems     |
| SellOrders         |
| Sell_Daily         |
| Suppliers          |
| Waters             |
| Workers            |
+--------------------+
```

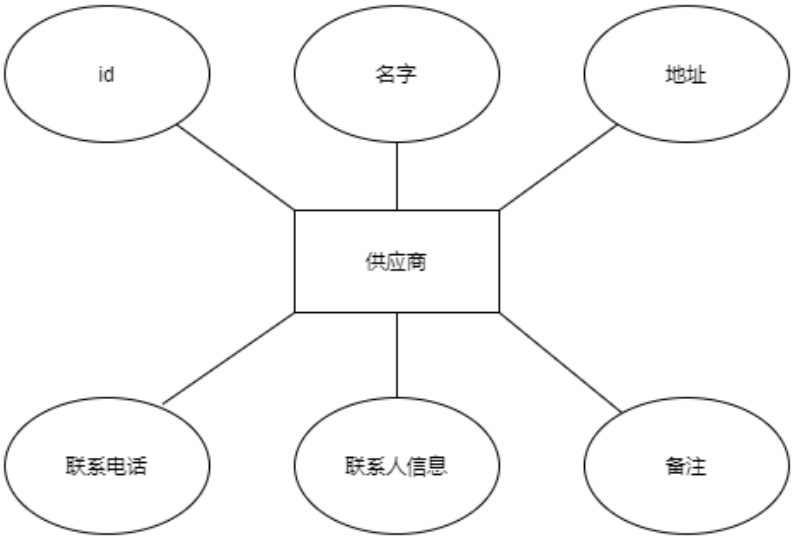
共有 8 张表，其中 Sell_Daily 和 Purchase_Daily 是视图。

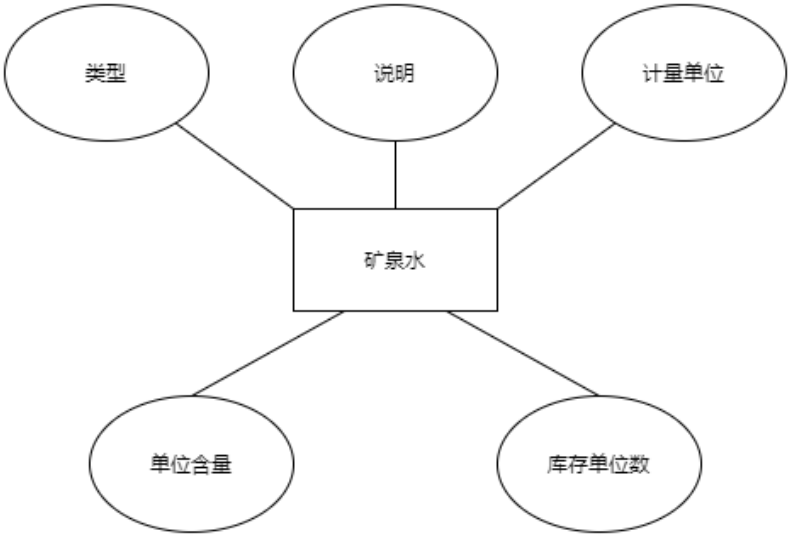
8 张表分别为：顾客，供应商，送水工，矿泉水，入库订单，入库订单条目，出库订单，出库订单条目。

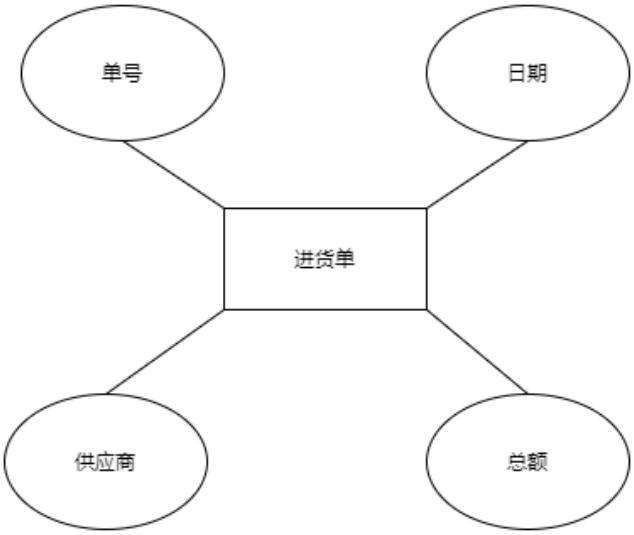
4.1 ER 图

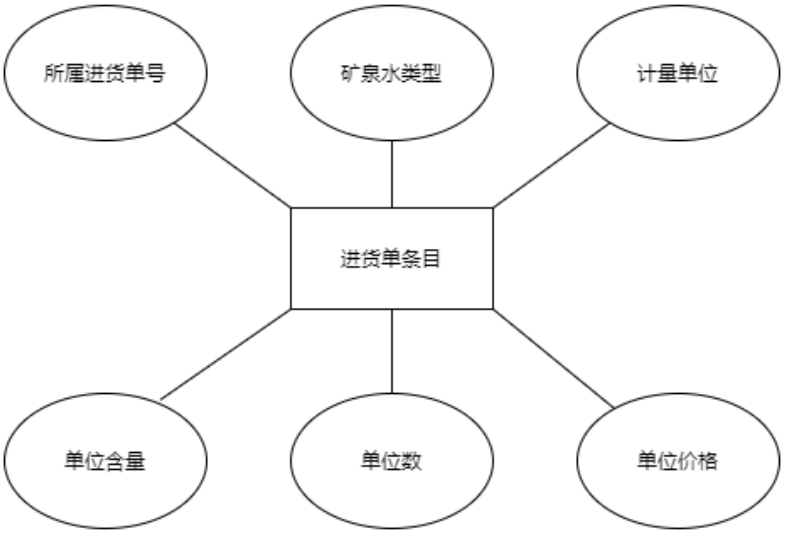


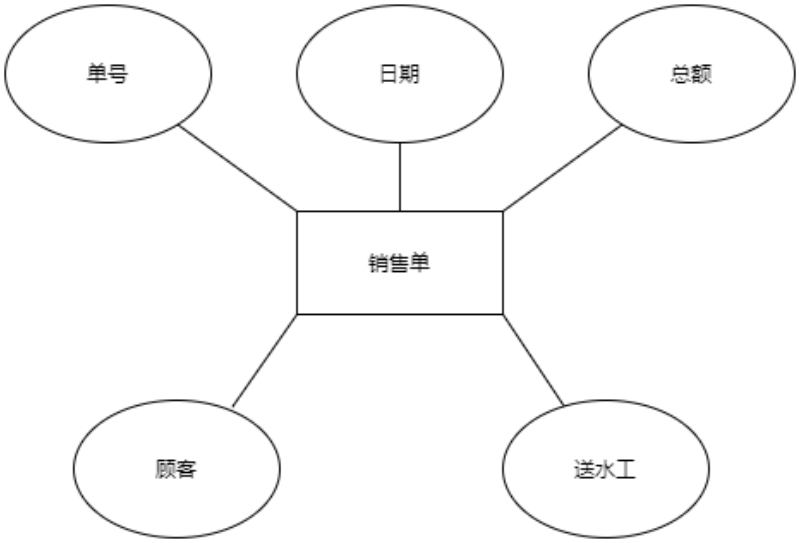

id
姓名
电话
顾客
住址
备注

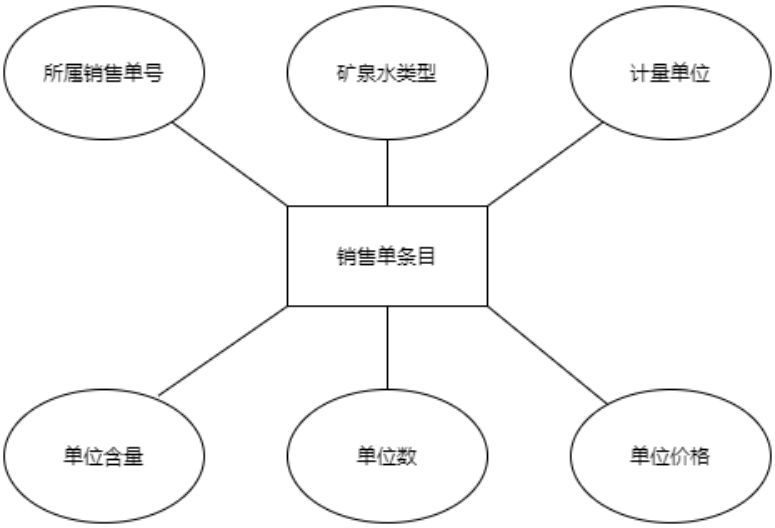

id
名字
地址
供应商
联系电话
联系人信息
备注

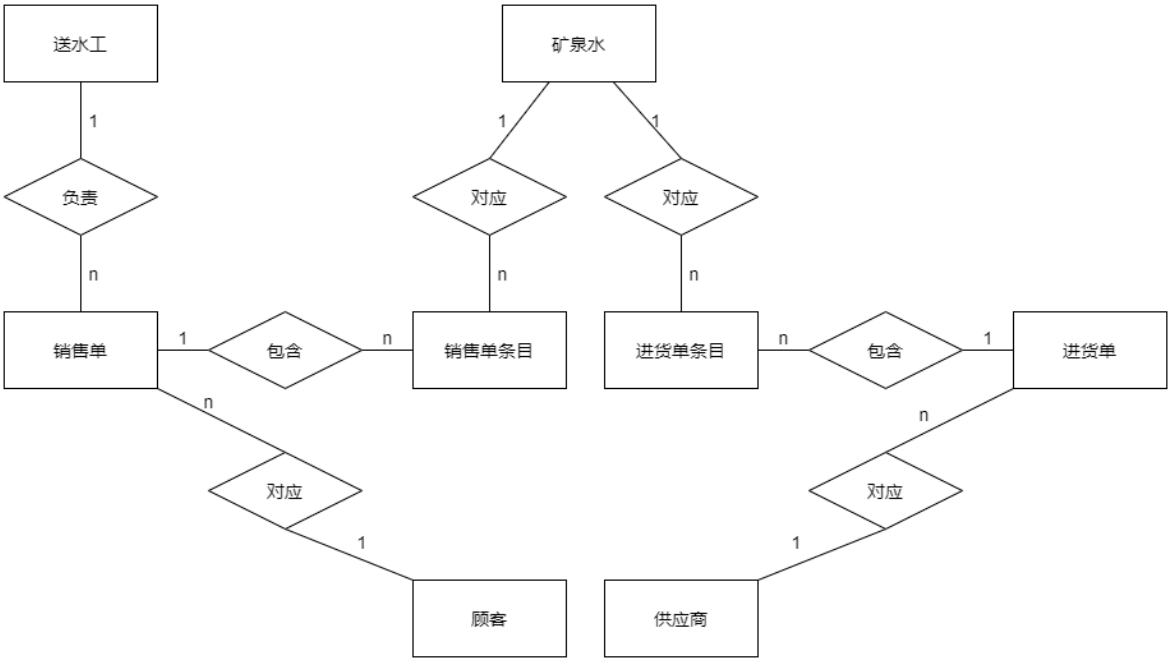

类型
说明
计量单位
矿泉水
单位含量
库存单位数


单号
日期
进货单
供应商
总额


所属进货单号
矿泉水类型
计量单位
进货单条目
单位含量
单位数
单位价格


单号
日期
总额
销售单
顾客
送水工





4.2 数据字典

送水工：

字段名	字段类型	允许为空	外键	备注
id	int	N		工号，主键，自增
name	varchar(32)	N		
gender	tinyint	N		0 为女性，1 为男性

age	tinyint unsigned	N		
hiredate	date	N		
phone	varchar(32)	N		

顾客：

字段名	字段类型	允许为空	外键	备注
id	int	N		主键，自增
name	varchar(32)	N		
phone	varchar(32)	N		
address	varchar(128)	N		
note	varchar(128)	Y		

供应商：

字段名	字段类型	允许为空	外键	备注
id	int	N		主键，自增
name	varchar(32)	N		
address	varchar(128)	N		
phone	varchar(32)	N		
contact_person	varchar(32)	N		
note	varchar(128)	Y		

矿泉水：

字段名	字段类型	允许为空	外键	备注
type	varchar(32)	N		主键
note	varchar(128)	Y		
unit_type	varchar(32)	N		桶/瓶，默认值“桶”

unit_amount	decimal(8,3)	N		单位含量以升计算
amount	int unsigned	N		默认值 0

进货单：

字段名	字段类型	允许为空	外键	备注
id	int	N		单号，主键，自增
date	date	N		
supplier_id	int	N	供应商 id	
price	decimal(8,2)	N		

进货单条目：

字段名	字段类型	允许为空	外键	备注
order_id	int	N		所属进货单号
water_type	varchar(32)	N	矿泉水 type	
unit_type	varchar(32)	N		计量单位
unit_amount	decimal(8,3)	N		单位含量
amount	int	N		单位数
unit_price	decimal(8,2)	N		单位价格

销售单：

字段名	字段类型	允许为空	外键	备注
id	int	N		单号，主键，自增
date	date	N		
supplier_id	int	N	供应商 id	
worker_id	int	N	送水工 id	
price	decimal(8,2)	N		

销售单条目：

字段名	字段类型	允许为空	外键	备注
order_id	int	N		所属销售单号
water_type	varchar(32)	N	矿泉水 type	
unit_type	varchar(32)	N		计量单位
unit_amount	decimal(8,3)	N		单位含量
amount	int	N		单位数
unit_price	decimal(8,2)	N		单位价格

4.3 关系表

```
create table Workers (
    id int auto_increment primary key,
    name varchar(20) not null
);

create table Waters (
    water_type int primary key,
    amount int unsigned default 0,
    description varchar(20) not null
);

create table Customers (
    id int auto_increment primary key,
    name varchar(20) not null,
    phone varchar(20) not null,
    address varchar(20) not null
);

create table Suppliers (
    id int auto_increment primary key,
    name varchar(20) not null unique
```

```
);

create table SellOrders (
    id int auto_increment primary key,
    order_date date not null,
    amount int nunsigned not null,
    price decimal(8,2) not null,
    customer_id int,
    worker_id int,
    foreign key (customer_id) references Customers (id),
    foreign key (worker_id) references Workers (id)
);

create table SellOrderItems (
    sell_order_id int not null,
    water_type int,
    amount int nunsigned not null,
    unit_price decimal(4,2) not null,
    foreign key (water_type) references Waters (water_type)
);

create table PurchaseOrders (
    id int auto_increment primary key,
    order_date date not null,
    amount int nunsigned not null,
    price decimal(8,2) not null,
    supplier_id int,
    foreign key (supplier_id) references Suppliers (id)
);

create table PurchaseOrderItems (
    purchase_order_id int not null,
    water_type int,
    amount int nunsigned not null,
    unit_price decimal(4,2) not null,
    foreign key (water_type) references Waters (water_type)
);
```

五、数据库物理设计

5.1 触发器

录入进货单和销售单时，应当自动将该单涉及的所有类型矿泉水的库存增加或减少。

```
create trigger purchase_trigger
after insert on PurchaseOrderItems for each row
update Waters set amount=amount+new.amount where water_type=new.water_type;

create trigger sell_trigger
after insert on SellOrderItems for each row
update Waters set amount=amount-new.amount where water_type=new.water_type;
```

5.2 存储过程

系统需要支持查看指定月份所有送水工的送水总量，以及指定月份顾客的订水量排名。为此，我们分别创建两个存储过程，其他类似的统计需求实现方法也类似。

```
delimiter $$
create procedure workers_monthly(in delta int) begin
   select w.name as name, w.id as id, a.amount as amount from (
      select worker_id, sum(amount) as amount from SellOrders
      where date_format(order_date, '%Y %m') =
         date_format(date_sub(curdate(), INTERVAL delta MONTH),'%Y %m')
      group by worker_id
   ) as a join Workers w on a.worker_id = w.id;
end$$
delimiter ;
```

该存储过程的入口参数为一个整数值 delta，为 0 则代表统计当月的情况，为 1 则统计上一月份，以此类推。

对该存储过程，我们进行了一点优化。我们并不是直接让销售单表和送水工表做内

连接，然后再做分组求和；而是先在订单销售表上做分组求和，再与送水工表做内连接。因为订单表会随着时间而显著增大，这样的顺序可以减小笛卡尔积的大小和时间开销。

另外一个存储过程同理。

```
delimiter $$
create procedure customers_monthly(in delta int) begin
   select c.name as name, c.id as id, a.amount as amount from (
      select customer_id, sum(amount) as amount from SellOrders
      where date_format(order_date, '%Y %m') =
         date_format(date_sub(curdate(), INTERVAL delta MONTH),'%Y %m')
      group by customer_id
   ) as a join Customers c on a.customer_id = c.id
   order by amount desc;
end$$
delimiter ;
```

5.3 视图

尽管基本任务要求中没有涉及到视图，但为了扩展系统，以适应实际使用需要，以及为了展示系统的可扩展性，我们设计了统计功能，将在首页通过图表展示指定时间段内的每日进销金额。

我们通过创建视图来实现该统计功能，其他类似的统计功能也可以用同样的手段进行实现。

```
create view Purchase_Daily as
select order_date, sum(price) as price from PurchaseOrders
group by order_date order by order_date desc;

create view Sell_Daily as
select order_date, sum(price) as price from SellOrders
group by order_date order by order_date desc;
```

5.4 数据完整性

5.4.1 DEFAULT 约束

在管理矿泉水时，新创建的矿泉水类型的库存单位数默认为 0。

5.4.2 CHECK 约束

所有出现价格的地方，数据列的类型为 decimal，小数点后两位，并且加了 CHECK 约束，以保证价格为非负数。

所有出现数量的地方，数据类型均设为了 unsigned，插入数据时或修改库存单位数时将保证单位数为非负数。

5.4.3 外键约束

为了保证引用完整性，我们在大多数有关联的表之间设置了外键。例如销售单中的送水工工号和顾客 id，销售单条目中的矿泉水类型。这样可以让数据库管理系统来帮助我们保证数据的一致性。

但出于性能考虑，在目前的数据库中，我们有两处地方没有使用外键，分别是进货条目中的“所属进货单号”和销售条目中的“所属销售单号”。

这是因为，随时系统使用时间的增长，销售单和进货单表将会变得非常大，如果每次创建条目，都需要数据库去检索订单表，确保单号存在，将会极大降低我们的系统性能。第二个原因是，每次插入的订单条目所对应的订单号，其实是插入对应的订单时系统生成的，而不是管理员手动输入的，我们不必担心数据的一致性。

5.4.4 其他机制

录入订单时，可能会产生数据完整性的问题。例如，向订单条目表中插入一个订单的各个条目时，如果其中一两个条目在插入时出现问题，而其他条目被正常创建，就会导致数据不完整、不正确。因此，在插入一个新订单时，数据库管理系统将开启一个事务，如果所有的条目都被成功插入，则事务将被 commit，否则将进行 rollback，并报告错误，这样就不会发生订单的部分信息被录入的情况。

同时，前后端的实现中也考虑到了数据的完整性。管理员在录入各种数据时，前端就将进行检查数据的格式等各种问题，尽量确保不会发送包含错误数据的请求至后端。例如，在创建一个订单时，前端将检查条目是否有重复，检查价格是否符合格式要求，等等。

六、应用程序设计

6.1 功能模块

6.1.1 登录模块

权限验证与安全性是非常重要的，可以说是一个管理系统一开始就必须考虑和搭建的基础核心功能。

虽然目前我们的系统的使用者只有公司管理员，系统的所有功能都应该开放给管理员。但是，为了增强系统的可扩展性，以在后期需求变更后能快速地修改系统，我们为系统增加了权限管理功能。我们所要做到的是：不同的权限对应着不同的路由，同时侧边栏也需根据不同的权限，异步生成。实现登录和权限验证的思路如下：

- 登录：当用户填写完账号和密码后向服务端验证是否正确，验证通过之后，服务端会返回一个 token，拿到 token 之后（我会将这个 token 存贮到 cookie 中，保证刷新页面后能记住用户登录状态），前端会根据 token 再去拉取一个 user_info 的接口来获取用户的详细信息（如用户权限，用户名等等信息）。
- 权限验证：通过 token 获取用户对应的 role，动态根据用户的 role 算出其对应有权限的路由，通过 router.addRoutes 动态挂载这些路由。

上述所有的数据和操作都是通过 vuex 全局管理控制的。

关键代码体现在 router.js 路由表，以及 main.js 中对 vue-router 做的修改。

```
// router.js

import Vue from 'vue';
import Router from 'vue-router';

import Login from '../views/login/';
const dashboard = resolve => require(['../views/dashboard/index'], resolve);

//所有权限通用路由表
//如首页和登录页和一些不用权限的公用页面
export const constantRouterMap = [
  { path: '/login', component: Login },
  {
```

```
    path: '/',
    component: Layout,
    redirect: '/dashboard',
    name: '首页',
    children: [{ path: 'dashboard', component: dashboard }]
  },
]

//实例化 vue 的时候只挂载 constantRouter
export default new Router({
  routes: constantRouterMap
});

//异步挂载的路由
//动态需要根据权限加载的路由表
export const asyncRouterMap = [
  {
    path: '/workers',
    component: Layout,
    name: '送水工管理',
    meta: { role: ['admin'] },
  },
  { path: '*', redirect: '/404', hidden: true }
];
```

// main.js

```
router.beforeEach((to, from, next) => {
  if (store.getters.token) { // 判断是否有 token
    if (to.path === '/login') {
      next({ path: '/' });
    } else {
      if (store.getters.roles.length === 0) { // 判断当前用户是否已拉取完 user_info 信
息
        store.dispatch('GetInfo').then(res => { // 拉取 info
          const roles = res.data.role;
          store.dispatch('GenerateRoutes', { roles }).then(() => { // 生成可访问的路
由表
            router.addRoutes(store.getters.addRouters) // 动态添加可访问路由表
            next({ ...to, replace: true }) // hack 方法 确保 addRoutes 已完成 ,set the
replace: true so the navigation will not leave a history record
          })
        }).catch(err => {
```

```
          console.log(err);
        });
      } else {
        next() //当有用户权限的时候，说明所有可访问路由已生成 如访问没权限的全面会自动进入
404 页面
      }
    }
  } else {
    if (whiteList.indexOf(to.path) !== -1) { // 在免登录白名单，直接进入
      next();
    } else {
      next('/login'); // 否则全部重定向到登录页
    }
  }
});
```

通过上述代码可以清晰地看到，系统的实现满足前文描述的安全性的要求，例如，未登录而直接使用 URL 地址将无法进入系统，并且会被重定向到登录页。

6.1.2 信息管理模块

管理模块的实现原理已在前文做过介绍。以顾客管理为例，流程如下：

页面进入顾客管理页面时，前端发送 get 请求到后端，后端执行相应 sql 语句，拿到数据后返回给前端，前端以表格形式将数据处理后显示在页面上；

页面上还有修改和录入顾客的按钮，点击按钮将弹出表单，填写完信息后同样发送 post 或 put 请求至后端，后端执行相应 sql 语句，返回执行结果给前端，若成功，则前端刷新页面，若失败，则弹出提示。

6.1.3 费用管理与统计模块

我们采用 ECharts 来展示费用和其他统计信息。Echarts 全称 Enterprise Charts，商业级数据图表，是一个纯 Javascript 的图表库，能够流畅的运行在 PC 以及移动设备上，兼容当前绝大部分浏览器。ECharts 最初由百度团队开源，并于 2018 年初捐赠给 Apache 基金会，成为 ASF 孵化级项目。选择 ECharts 同样也是因为它有大量的参考资料以及完整的文档和用法示例。

这里对费用管理，也就是每日进销总额的展示的实现进行简单说明：用户选择好日

期区间后（默认为最近 7 天），以开始和结束日期作为请求的参数，发送两个 get 请求到后端，后端查询我们创建的两个视图（purchase_daily 和 sell_daily），返回两个数据到前端。由于并非每日都有进货或销售数据产生，前端需要对两个数组进行处理，没有订单的日期需要填充上 0 值，再计算每日收支，最后再图标上显示。

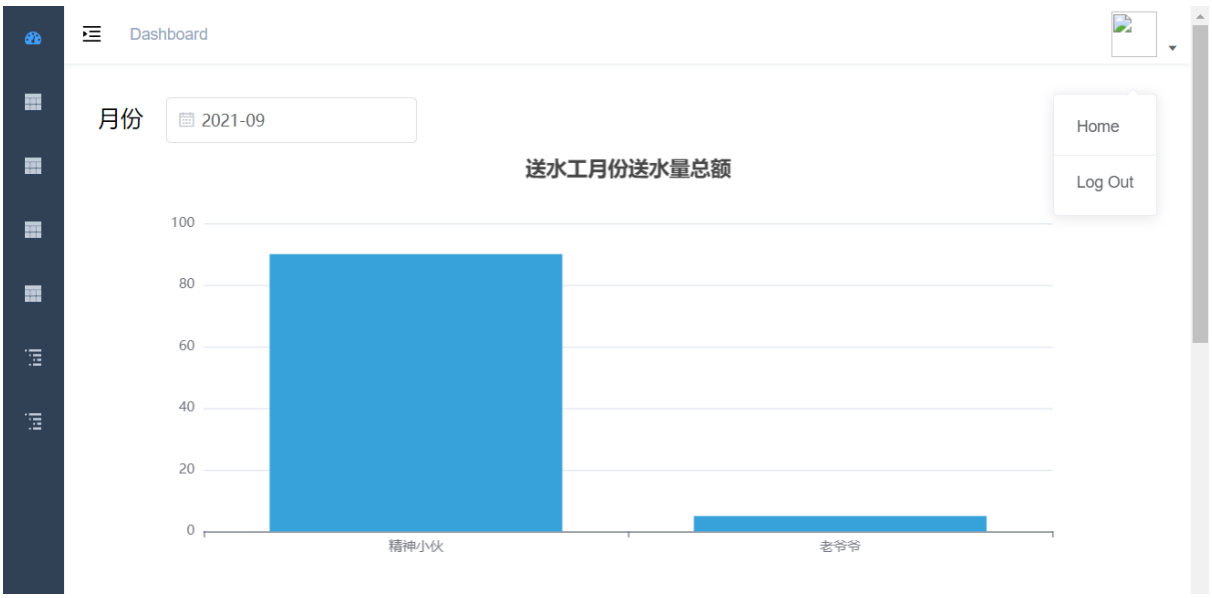
大致代码如下：

```
handleSpRangeChange() {
  const range =
    getBetweenDate(new Date(this.spRange[0]), new Date(this.spRange[1]))
  const purchaseData = (new Array(range.length)).fill(0)
  const sellData = (new Array(range.length)).fill(0)
  this.spBarOption.yAxis[0].data = range
  const start = yyyyMMdd(new Date(this.spRange[0]))
  const end = yyyyMMdd(new Date(this.spRange[1]))

  getPurchaseDaily(start, end).then(res => {
    res.data.forEach(e => {
      const date = yyyyMMdd(new Date(e.order_date))
      const index = range.indexOf(date)
      purchaseData[index] = -e.price
    })
    this.spBarOption.series[2].data = purchaseData

    return getSellDaily(start, end)
  }).then(res => {
    res.data.forEach(e => {
      const date = yyyyMMdd(new Date(e.order_date))
      const index = range.indexOf(date)
      sellData[index] = e.price
    })
    this.spBarOption.series[1].data = sellData

    let profitData = (new Array(range.length)).fill(0)
    profitData = profitData.map((v, i) => sellData[i] + purchaseData[i])
    this.spBarOption.series[0].data = profitData
  })
}
```

6.2 界面设计

6.2.1 总体设计

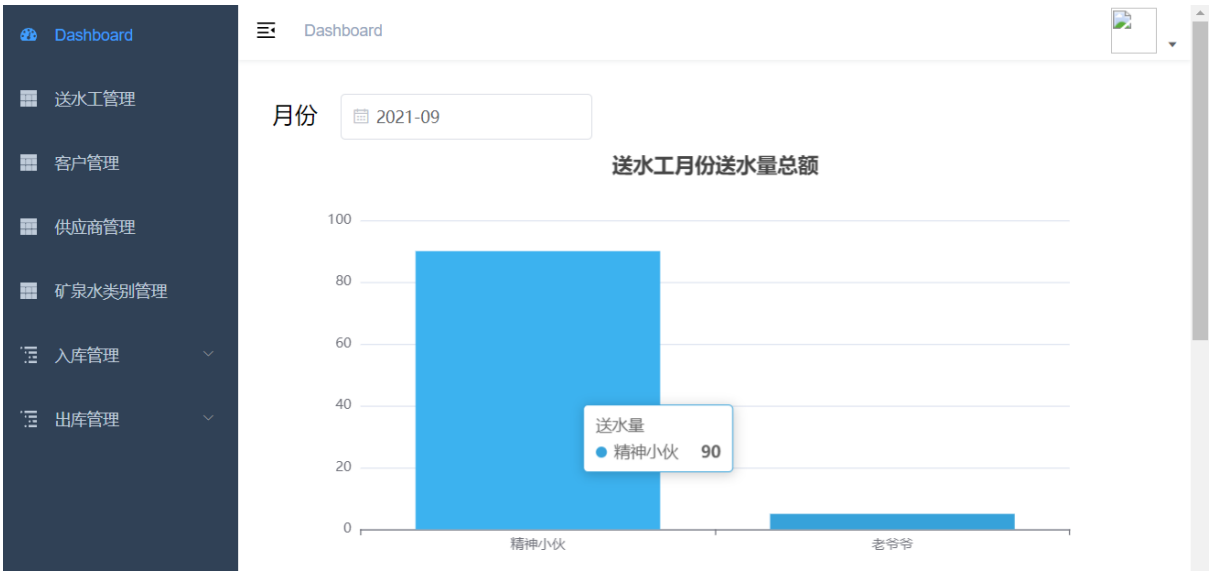
左侧为导航栏，导航栏可以收起，右侧为主体部分。顶部有状态栏，状态栏左侧显示当前位于哪个功能模块，点击右侧头像可以展开下拉菜单退出登录。

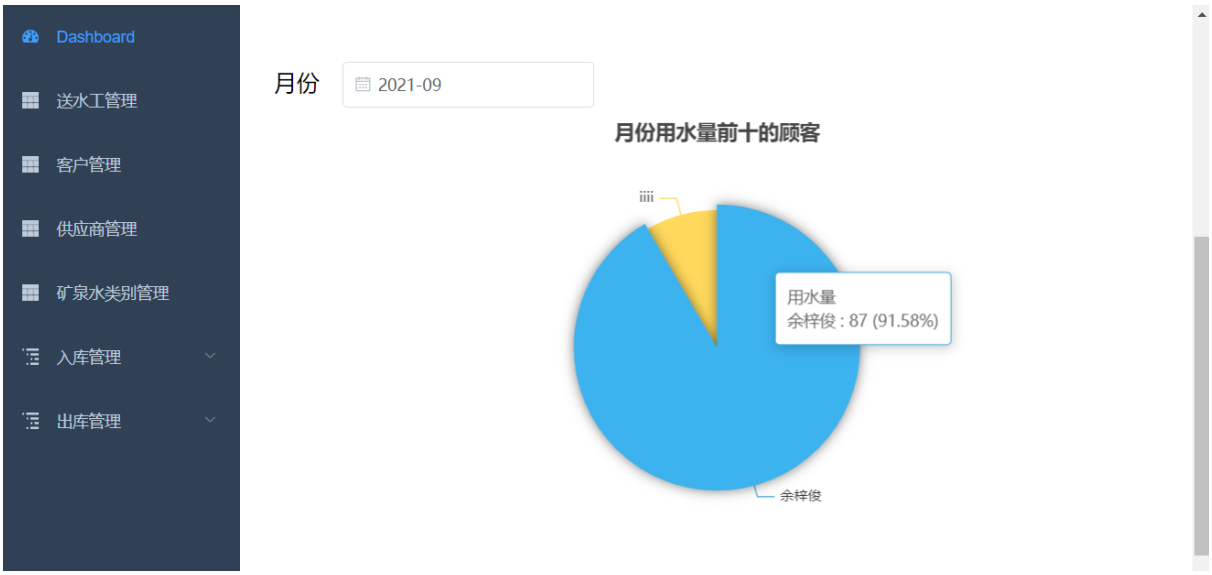


6.2.2 登录模块

6.2.3 信息管理模块

6.2.4 费用管理与统计模块






Dashboard
送水工管理
客户管理
供应商管理
矿泉水类别管理
入库管理
出库管理
日期区间
2021-09-23
至
2021-09-28
日收支
进水支出
卖水收入
2021-09-28
-67
-102
35
2021-09-27
0
0
0
2021-09-26
-30
-30
0
2021-09-25
305
-10
315
2021-09-24
-60
-60
0
2021-09-23
-100
-100
0
-200
-100
0
100
200
300
400
2021-09-28
日收支 -67
卖水收入 35
进水支出 -102


Dashboard
送水工管理
客户管理
供应商管理
矿泉水类别管理
入库管理
出库管理
日期区间
2021-09-23
至
2021-09-28
日收支
进水支出
卖水收入
2021-09-28
-102
2021-09-27
-30
2021-09-26
-10
2021-09-25
-60
2021-09-24
-100
2021-09-23
-200
-100
0
100
200
300
400
2021-09-28
日收支 -67
卖水收入 35
进水支出 -102

七、总结

在本次软件工程师综合训练中，我们以《数据库系统》和《软件工程》课程的相关知识为基础和中心，以团队合作的形式研发了一个完整的送水管理系统。实验中我们分工明确，组员认真负责，完成了数据库设计并做出了可用的交互界面。

该系统有三处亮点。一是采用了交互式的图表对费用和统计数据进行了可视化。系统也有良好的灵活性和可扩展性，虽然我们只做了简单的费用管理和可视化，但使用相同的原理，该系统能方便地增加更多功能。二是具有较高的安全性，拥有用户登录和权限控制功能，能拒绝错误的操作和数据。在胡老师的提醒过后，我们进一步完善了对操作和数据的检查。三是没有完全依靠数据库来保证数据的完整性。我们考虑到了实际的使用情况，在可能会出现性能问题的地方舍弃了对外键的使用。

而系统最大的问题则在于需求分析的不透彻，不完整以及不严谨。在一开始，我们就欠缺对该系统实际使用场景的考量。我们只是通过实验指导书的题目描述作为参考来实现我们的系统，但却没有考虑到实际情况的复杂性，例如胡老师指出，送水公司的订单通常会写某种水的件数和规格等，而我们的系统最初只有数量，没有计量单位和规格等字段。没有完整的需求分析，就导致了我们的系统很难有实际的生产使用价值。

我们认真听取了胡老师在验收时所给的建议，进一步思考、讨论、完善了系统的设计，但由于时间和能力有限，我们的系统实现没能进行全面的完善。本次综合训练使我明白了要做好一个系统，应当遵循软件工程方法学，不能漏掉工程的任何一环，并且要结合实际情况来设计数据和系统。

附. 参考文献

[1] 王珊, 萨师煊. 数据库系统概论[M]. 第 5 版. 高等教育出版社, 2014.

[2] 张海藩, 牟永敏. 软件工程导论[M]. 第 6 版. 清华大学出版社, 2013.

[3] MySQL 中文网. MySQL 中文文档[EB/OL]. [2020-10-19]. https://www.mysqlzh.com/.

[4] Vue 开发团队. Vue.js Guide[EB/OL]. [2021-10-19]. https://cn.vuejs.org/v2/guide/.

[5] Vue 开发团队. Vue Router Guide[EB/OL]. [2021-10-19]. https://router.vuejs.org/zh/guide/.

[6] Vue 开发团队. Vuex Guide [EB/OL]. [2021-10-19]. https://vuex.vuejs.org/zh/guide/.

[7] Apache 基金会. Apache ECharts 使用手册[EB/OL]. [2021-10-19]. https://echarts.apache.org/handbook/zh/get-started/.

[8] Node.js 官方网站. Node.js 入门教程[EB/OL]. [2021-10-19]. http://nodejs.cn/learn.

[9] Express 官方网站. Express Guide[EB/OL]. [2021-10-19]. http://expressjs.com/en/guide/routing.html.

[10]RESTful CN 网站. RESTful API 设计风格[EB/OL]. [2021-10-19]. https://restfulapi.cn/.

[11]PanJiaChen. 手摸手，带你用 vue 撸后台 系列二(登录权限篇)[EB/OL]. [2020-10-19]. https://juejin.cn/post/6844903478880370701.