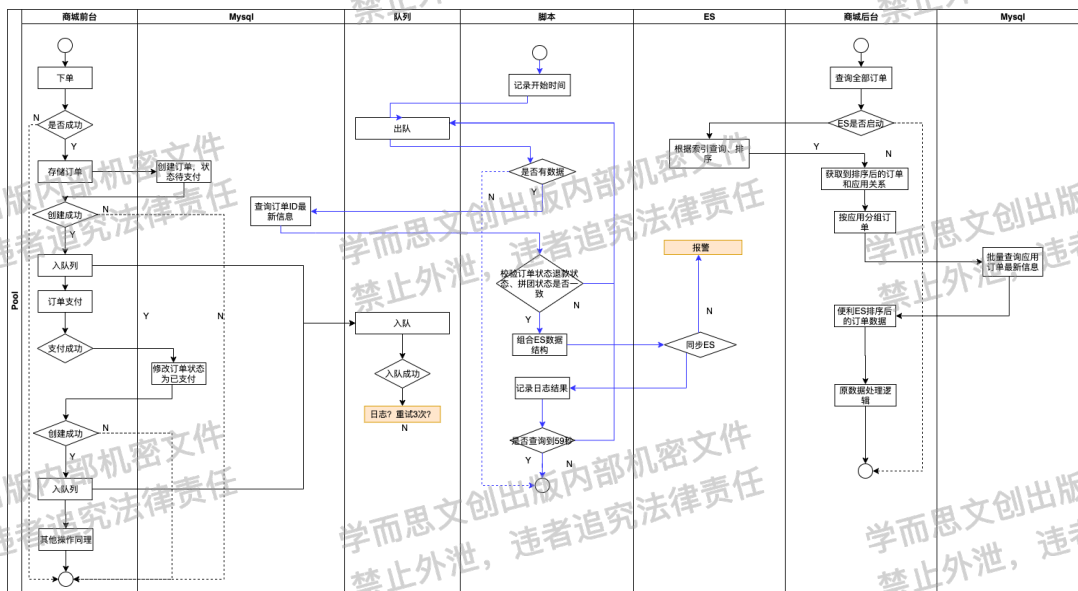


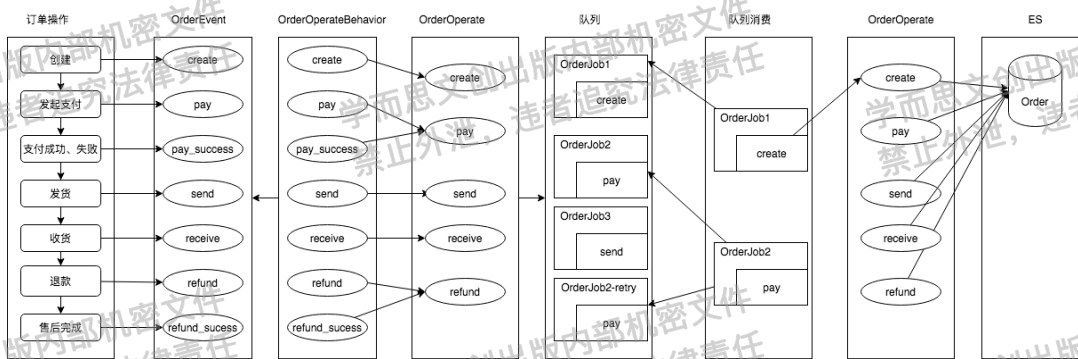
ES订单整合

一、订单流程：

订单操作 -> 触发订单事件 -> 生成队列任务 -> 入队列
队列消费 -> 解析队列任务 -> 进行消费同步ES库 -> 错误处理
查询操作 -> ES开关 -> ES搜索 -> 数据库查询 -> 接口返回



二、订单事件及入队过程



三、队列服务实现



四、事件触发

1、添加事件

```

function behaviors()
{
    $behaviors=parent::behaviors();
    $behaviors['log']=LogBehavior::class;
    $behaviors['order_operate']=OrderOperateBehavior::class;
    return $behaviors;
}
    
```

2、触发事件

```

//v1.3.1 订单创建成功 分发事件 可以处理不印象业务结果的
$this->trigger( name: OrderEvent::EVENT_AFTER_CREATE,new OrderEvent([
    'orderSn'=>$orderInfo['order_sn'],
    'productId'=>$this->productId
]));
    
```

3、事件监听

```

class OrderOperateBehavior extends Behavior
{
    function events()
    {
        return [
            OrderEvent::EVENT_AFTER_CREATE => "eventAfterCreate", // 添加订单成功
            OrderEvent::EVENT_BEFORE_PAY => "eventBeforePay", // 支付成功付成功
            OrderEvent::EVENT_PAY_SUCCESS => "eventPaySuccess", // 支付成功付成功
            OrderEvent::EVENT_CHANGE_ADDRESS => "eventChangeAddress", // 支付成功付成功
            OrderEvent::EVENT_AFTER_SEND => "eventAfterSend", // 发货后
            OrderEvent::EVENT_AFTER_RECEIVE => "eventAfterReceive", // 收货之后
            OrderEvent::EVENT_AFTER_CLOSE => "eventAfterClose", // 关闭订单
            OrderEvent::EVENT_AFTER_REFUND => "eventAfterRefund", // 申请售后之后
            OrderEvent::EVENT_REFUND_CLOSE => "eventRefundClose", // 售后结束
            OrderEvent::EVENT_REFUND_FINISHED => "eventRefundFinished", // 售后成功
        ];
    }

    function attach($owner)
    {
        parent::attach($owner); // TODO: Change the autogenerated stub
    }

    /**
     * 订单创建成功 处理逻辑
     *
     * @param $event
     */
    function eventAfterCreate(OrderEvent $event)
    {
        if (empty($event->orderInfo['user_id'])) {
            return false;
        }
        $this->addEsOrderQueue($type, OrderOperateConst::OPERATE_ORDER_CREAT, $event->orderInfo);
    }
}

```

五、入队

```

//初始化ES队列任务
$jobData = new OrderEsJob([
    'operateType' => $type,
    'userId' => intval($orderInfo['user_id']),
    'orderSn' => $orderInfo['order_sn'],
    'orderStatus' => intval($orderInfo['order_status']),
    'orderTime' => intval($orderInfo['created_at']),
    'productId' => intval($orderInfo['butt_product_id']),
]);
$status = QueueService::push($jobData, maxTime: 3);

if (!$status) {
    //入队不成功 不影响用户操作
} else {
    //成功暂不处理 入队成功 异步同步ES 可能会出现ES同步失败的情况 后期有可能会改为同步操作
}

```

六、队列消费

```

while (time() - $startTime < 60) {
    $jobData = '';
    try {
        $jobData = QueueService::pop( key: QueueConst::ES_ORDER_MERGE_QUEUE);
        if (empty($jobData)) {
            sleep( seconds: 1);
            continue;
        }
        $result++;
        list($status, $jobData) = $logic->dealEsOrder(QueueService::handle($jobData));
        if ($status) {
            $message.= $jobData->orderSn. ',';
            $success++;
        } else {
            QueueService::retry($jobData);
        }
    } catch (\Exception $e) {
        \Yii::error([
            'logKey' => 'es_order_read_error',
            'logMsg' => debugError($e),
            'logData' => $jobData,
        ]);
    }
}

```

学而思文创出版内部机密文件
禁止外泄，违者追究法律责任

学而思文创出版内部机密文件
禁止外泄，违者追究法律责任

学而思文创出版内部机密文件
禁止外泄，违者追究法律责任

学而思文创出版内部机密文件
禁止外泄，违者追究法律责任

学而思文创出版内部机密文件
禁止外泄，违者追究法律责任

学而思文创出版内部机密文件
禁止外泄，违者追究法律责任

学而思文创出版内部机密文件
禁止外泄，违者追究法律责任

学而思文创出版内部机密文件
禁止外泄，违者追究法律责任

学而思文创出版内部机密文件
禁止外泄，违者追究法律责任

学而思文创出版内部机密文件
禁止外泄，违者追究法律责任

学而思文创出版内部机密文件
禁止外泄，违者追究法律责任

学而思文创出版内部机密文件
禁止外泄，违者追究法律责任

学而思文创出版内部机密文件
禁止外泄，违者追究法律责任

学而思出版内部机密文件
禁止外泄，违者追究法律责任

学而思文创出版内部机密文件
禁止外泄，违者追究法律责任

学而思文创出版内部机密文件
禁止外泄，违者追究法律责任

学而思文创出版内部机密文件
禁止外泄，违者追究法律责任

学而思文创出版内部机密文件
禁止外泄，违者追究法律责任

学而思文创出版内部机密文件
禁止外泄，违者追究法律责任

学而思文创出版内部机密文件
禁止外泄，违者追究法律责任

学而思文创出版内部机密文件
禁止外泄，违者追究法律责任