2022-05-07_战区工作台二期看板卡片 前端UI组件设计

Created and last modified by 王文磊 on May 18, 2022

项目结构

```
.src/componet
  - RnCharts （引入图表）
HomePage
  - CardDashboard（卡片看板）
    - util
      - CardSorting （图表排序）
    - Card（卡片）
      - util
        - CardAdapter （图表数据结构适配器）
          - BaseAdapter.ts
          - PieChartAdapter.ts
          - LineChartAdapter.ts
          - RadarChartAdapter.ts
      - header.tsx
      - content.tsx
      - index.tsx
      - store.ts
      - style.ts
    - FullScreenCard（全屏卡片）
      - header.tsx
      - index.tsx
      - style.ts
    - const.ts（图表组件静态配置）
    - index.tsx
    - store.ts
  - index.tsx （首页入口）
  - store.ts
```

功能拆解

功能	描述	重点
看板加载	同步PC端驾驶舱公共概览的看板信息，获取该看板中的所有图表的基础信息，做排序，生成若干卡片	排序
图表加载	根据看板中获取到的cardId加载对应的卡片，应用不同的策略挂载不同的组件，适配数据	数据适配 状态判断： 1. 可支持 -饼图 -折线图 -雷达图 2. 无权限的 展示无权限缺省图 3. 不支持的图表类型 展示暂不支持缺省图
指标说明	全局的指标说明popup	popup组件

功能	描述	重点
进入全屏	复杂图表类型的卡片需要支持横屏查看	react-native-charts-wrapper横屏支持

一、首页入口

1. **UI部分**
   该部分由weiduo.chen开发，将封装好的 CardDashboard 组件与 PopUp 添加至下图指示位置

```
public render() {
  const { loadAllTaskList, loading } = store;
  return (
    <Page
      statusBarADRBgColor={{ color: '#000' }}
      style={styles.page}
      ipxBottomStatusBar={false}
    >
      <NavBar
        title="战区工作台"
        style={styles.navbar}
        leftBtnStyle={styles.navLeftBtn}
        segmentationLine={false}
      />
      <ScrollView
        style={styles.container}
        refreshControl={
          <RefreshControl
            refreshing={loading}
            onRefresh={loadAllTaskList}
          />
        }
      >
        <BasicInfo />
        <TaskPool />

                    于此处添加CardDashboard与PopUp

      </ScrollView>
    </Page>
  )
}
```

2. **state部分**

   在HomePage最外层的store中需要增加 popUpVisible 控制 popUp 展示

```
interface IHomePageStore {
  ... // 现有逻辑
  popupVisible: boolean; // popUp展示控制
  setPopUpVisible: (visible: boolean) => void; // 修改popUpVisible
};
```

二、卡片看板

多个卡片的父组件，需要做的是根据 chartsBaseInfoList 渲染对应数量的卡片

卡片顺序及数据结构的处理

挂载全屏节点，根据传递给Card的回调触发时传递的id全屏展示相应的图表

设计稿：


B01-首页
战区工作台
我的概览
数据时效：2022年12月12日 12:00:00
编辑概览
逾期任务 >
139 项
逾期总数
P1任务：12项
P2任务：4项
P3任务：123项
立即处理 >
2 项
P1
今日完成 >
21 项
P2
近期完成 >
124 项
P3
雷达图示例：运营量化评估-店铺三支柱模型
卡片看板区域
能力指数
难度系数
意愿指数
首页
清洁计划
问题整改
门店巡查
折线图示例：商品津贴管理费

输入：

```
interface ICardDashboard {
  showPopUp: () => void; // 展示popup
};
```

1. **CardDashboardStore**

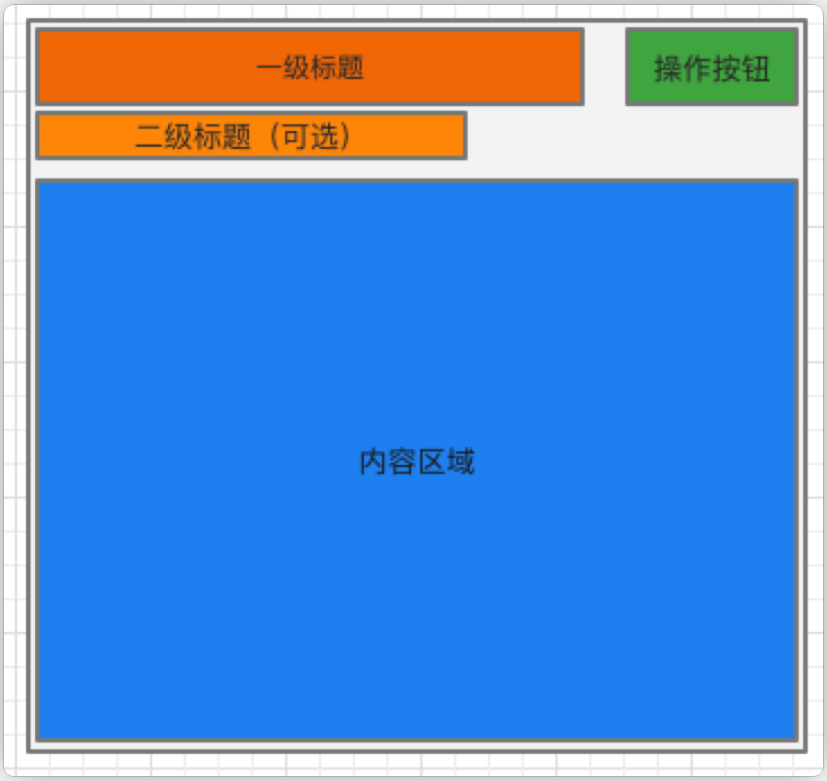
```
interface ICardDashboardStore {
  fullScreenVisible: boolean; // 是否展示全屏卡片
  setFullScreenVisible: (visible: boolean) => void; // 修改全屏卡片可见性
  chartsBaseInfoList: []; // 所有图表基础信息数组，提供给 CardDashboard 遍历提取id、title等渲染C
  fetchChartsBaseInfo: () => void; // 请求接口获取所有用户可见的图表基础信息，更新chartsInfoList
};
```

   在store init 与 首页组件 RefreshControl onRefresh 时需要调用 fetchChartsBaseInfo() 方法获取看板信息
2. **图表排序**
   看板图表排序方法，按可展示 > 无权限 > 不支持 排序
3. **全屏卡片**

```
interface IFullScreenCard {
  visible: boolean; // 是否可见
  onClose: () => void; // 关闭回调
  chartInfo: {
    id: number; // 图表id
    chartType: string; // 图表类型
  };
  data: {}; // 使用适配器转换后的图表数据
};
```

4. **Const**
   维护已有图表的静态props，与data结合后传递给封装好的图表组件

三、卡片



输入：

```
interface IProps {
  chartInfo: {
    id: number; // 图表id
    chartType: string; // 图表类型
  };
  data: []; // 适配器转换后的数据
  enterFullScreen: (id: number) => void; // 点击全屏的回调
  showPopUp: () => void; // 展示popup
};
```

功能：

1. **接收卡片看板传递的chartType判断是否需要请求图表信息，对于一些不支持的图表类型内容部分直接返回缺省图，对于支持的图表则需要请求图表数据并映射至相关图表组件**
2. **适配器**
   输入：从接口获取的原始数据
   输出：可直接被图表组件使用的数据
   BaseAdapater：抽离公共的父类，后续增加其他图表适配器时继承
   由于技术框架的不同，pc上图表的数据结构无法兼容至移动端，因此需要对基于 react-native-charts-wrapper 封装的图表组件做一层数据适配
3. **一级、二级标题展示**
4. **操作按钮：全局popup指标说明、全屏查看当前图表**
5. **自适应内容区域：展示不同图表内容**

header

输入：

```
interface ICardHeader {
  title: string; // 标题
  subTitle?: string; // 二级标题
  enterFullScreen: (id: number) => void; // 点击全屏的回调
  showPopUp: () => void; // 展示popup
};
```

content

输入：

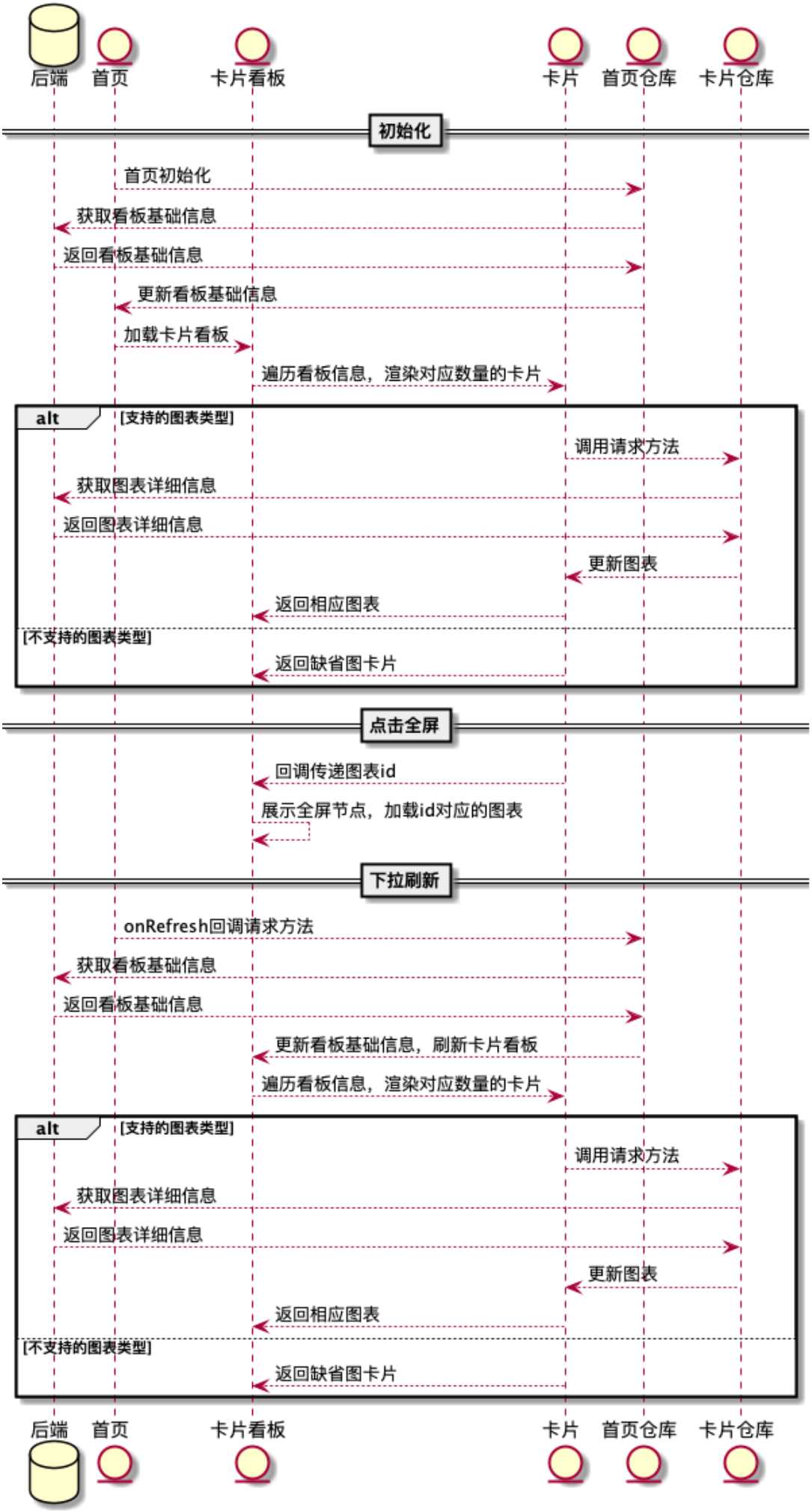
```
interface ICardContent {
  chartType: string; // 图表类型
  data: []; // 适配器转换后的数据
};
```

map图表：

```
constructor() {
    this.chartMap = {
        'pieChart': (data) => <PieChart data={data} ...pieChartStaticProps />,
        'lineChart': (data) => <LineChart data={data} ...lineChartStaticProps />,
        ...,
        'authless': () => <Authless />,
        'unsupport': () => <UnSupport />,
    }
}

getChart = (chartType, data) => {
    return this.chartMap[chartType](data);
}
```

四、项目流程


后端
首页
卡片看板
卡片
首页仓库
卡片仓库
初始化
首页初始化
获取看板基础信息
返回看板基础信息
更新看板基础信息
加载卡片看板
遍历看板信息，渲染对应数量的卡片
alt
[支持的图表类型]
调用请求方法
获取图表详细信息
返回图表详细信息
更新图表
返回相应图表
[不支持的图表类型]
返回缺省图卡片
点击全屏
回调传递图表id
展示全屏节点，加载id对应的图表
下拉刷新
onRefresh回调请求方法
获取看板基础信息
返回看板基础信息
更新看板基础信息，刷新卡片看板
遍历看板信息，渲染对应数量的卡片
alt
[支持的图表类型]
调用请求方法
获取图表详细信息
返回图表详细信息
更新图表
返回相应图表
[不支持的图表类型]
返回缺省图卡片
后端
首页
卡片看板
卡片
首页仓库
卡片仓库

五、排期

未开始：UNDO

完成状态：DONE

进行中：IN PROGRESS

	模块	功能点	问题点	工作量	Block原因（没有不填）	状态
1	首页	插入CardDashboard，PopUp组件	项目、BeesWax中是否有popup组件，若没有考虑使用其他组件替代	4h		DONE
2	卡片看板	整体布局，卡片排序，根据 chartsBaseInfoList 渲染对应的卡片，挂载全屏节点		6h		DONE
3	卡片	卡片样式、根据图表类型做相应渲染、数据适配、图表组件静态配置、全屏点击回调、popup回调	react-native-charts-wrapper是否支持横屏	10h		IN PROGRESS
4	公共方法	适配器：饼、雷达、折线图数据结构适配 卡片排序：已支持 > 无权限 > 未支持		5h		DONE
5	图表组件静态配置	已支持的图表组件，达成设计稿形态需要的静态配置项		3h		DONE
6	接口数据结构整理	整理驾驶舱目前接口的数据格式，由于后段资源没跟上，只能知道与PC驾驶舱数据结构相同，没有接口文档，需要自行整理		2h		DONE
5	自测			0.5d		UNDO
6	联调			1d		UNDO
	总计			42h		

Like Be the first to like this

评论

王宝同 发表：

0、缺少功能拆解：需要整体轮询二期部分需要实现的功能全集
1、Dashboard部分独立一个store，这部分和首页关系不大，在这期是设计到了一起，后续迭代可能会被拆开
2、内容部分，使用顶部项目结构用到的名称来叙述，上下要一致，避免类似【所有图表卡片的外层容器】这种模糊的描述；CardDashboard在项目结构里也没有体现
3、Card组件是否继续拆分?
4、在哪里维护已有展示组件信息?
5、全屏的设计方案，组件放在哪里?
6、popover的组件所在层级?
7、一次性加载所有是否合理，是否会遇到性能问题？看板加载过程是否有lazyload控制？如果有应该如何设计
8、图表是否需要排序: 可展示的/无权限的/系统不支持的？和产品确认
9、最顶部的代码结构，补充组件/class的具体api定义，在文档中的位置可以放到布局拆解的后面

```
HomePage
- CardGroup （卡片看板）
- util
- CardAdapter （图表数据结构适配器）
- 这里适合定义统一的BaseAdapter，定义接口信息
- PieChartAdapter extends BaseAdapter
transform(input: IInput): IPieChartData
- LineChartAdapter
- RadarChartAdapter
- Card（单个卡片）
- index.tsx
- 这里是否继续划分组件? toolbar, content
- enterFullScreen
- showPopover
- xxxxxx
- store.ts
功能点1
功能点2
......
- style.ts


- index.tsx
- index.tsx （首页入口）
- store.ts
功能点1
功能点2
......
```