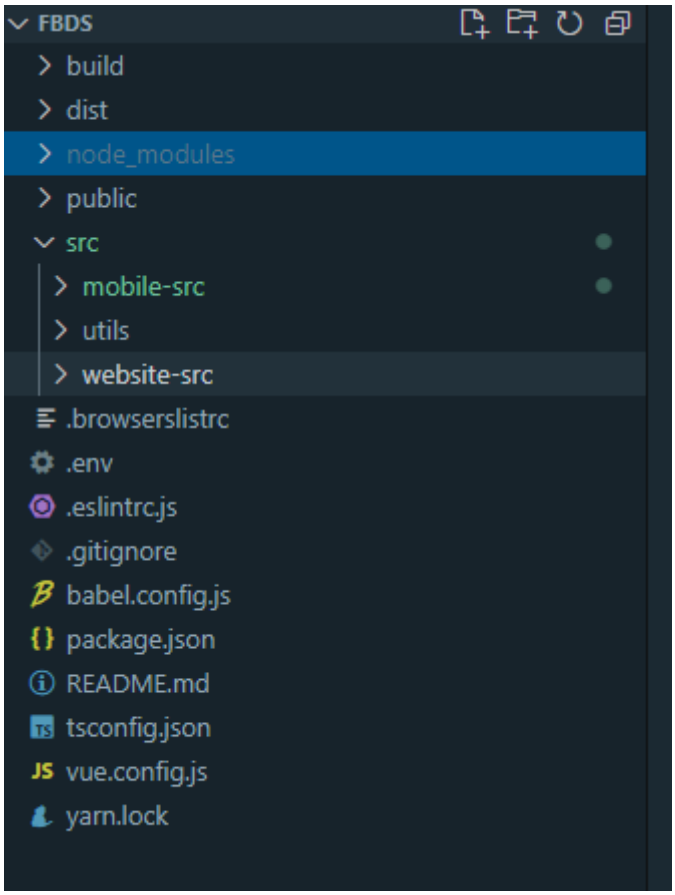
银行营销低代码平台

基础能力

1. ts+vue3基础技术+monorepo的管理项目代码方式
    - 项目分为编辑器以及演示器两个子项目。



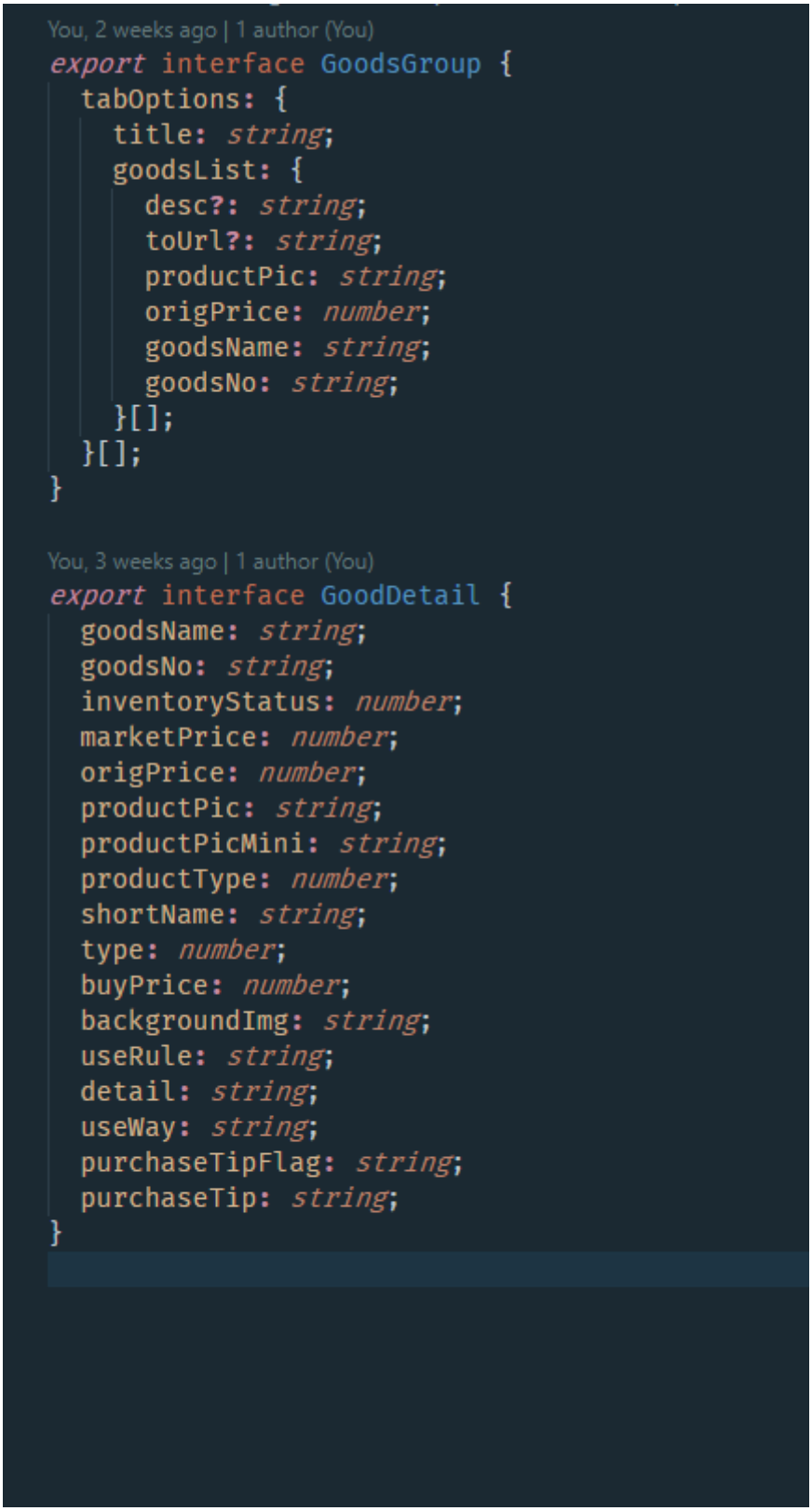
    - 灵活增减修改新组件，单一repo即可完成编辑器以及演示器的代码编辑。
    - 两个项目皆由ts+vue3完成，由公共文件编辑每个组件对应的tpyel以及json schema，保证两端组件的接口一致性。

      例子：商品导航分组的公共类型强制校验文件

      schema类型文件：

```ts
const schema: JSONSchemaType<GoodsGroup> = {
  $schema: "http://json-schema.org/draft-07/schema#",
  type: "object",
  properties: {
    tabOptions: {
      type: "array",
      items: {
        type: "object",
        properties: {
          title: {
            type: "string",
          },
          goodsList: {
            type: "array",
            items: {
              type: "object",
              properties: {      You, 3 weeks ago • feat: goodGroup
                desc: {
                  type: "string",
                  nullable: true,
                },
                shortName: {
                  type: "string",
                  nullable: true,
                },
                toUrl: {
                  type: "string",
                  nullable: true,
                },
                marketPrice: {
                  type: "number",
                  nullable: true,
                },
                productPic: {
                  type: "string",
                },
                origPrice: {
                  type: "number",
                },
                goodsName: {
                  type: "string",
                },
                goodsNo: {
                  type: "string",
                },
              },
              required: ["productPic", "origPrice", "goodsName", "goodsNo"],
              additionalProperties: false,
            },
          },
        },
        required: ["title", "goodsList"],
        additionalProperties: false,
      },
    },
  },
  required: ["tabOptions"],
  additionalProperties: false,
  definitions: {},
};
// validate is a type guard for MyData - type is inferred from schema type
export const validate = ajv.compile(schema);
```

type文件：



- 后续方便配置独立发布，为性能提升提供延展性。

2. 通过构建演示器与编辑器的发布响应式系统提供**通用的**可视化响应式编辑能力，抽离公共hook，不区分具体组件，对单个组件的所有数据进行监听。
    - 演示器：发布浮动层事件，订阅编辑器对组件数据修改的事件。

```
};
const handleMessage = (e: { data: PostMessageType }) => {      You, 2 m
  const { data } = e;
  switch (data.type) {
    case "addComponent":
      console.log(data.payload);
      nowComponents.value.push(data.payload);
      break;
    case "changeComponent":
      nowComponents.value[seletedComponentIndex.value] = data.payload;
      console.log("tag", nowComponents);
      break;
    case "savePage":
      postMessageToTop({
        type: "savePage",
        payload: nowComponents.value.map((v) => toRaw(v)),
      });
      break;
    default:
      break;
  }
};
useListener(handleMessage);
```

- 编辑器：发布组件数据修改的事件，订阅浮动层事件。

```
// 通信相关
const handleMessage = async (e: { data: PostMessageType }) => {
  const { data } = e;
  switch (data.type) {
    case "selectComponent":
      console.log("tag", data);
      drawer.componentConfigCode = data.payload.componentConfigCode;
      drawer.componentStyle = data.payload.componentStyle;
      drawer.componentConfigId = data.payload.componentConfigId;
      drawer.props = data.payload.props;
      drawer.ifOpen++;
      break;
    case "savePage":
      if (data.payload && data.payload.length) {
        const blockArr = data.payload.map((value) => {
          return {      You, a month ago • feat: connected
            componentConfigId: value.componentConfigId,
            componentConfigCode: value.componentConfigCode,
            componentStyle: JSON.stringify(value.componentStyle),
            props: JSON.stringify(value.props),
          };
        });
        console.log("tag", data.payload);
        const result = await putPage(nowRoute.query.id, {
          title: titleConfig.value,
          blocks: blockArr,
        });
        if (result) {
          message.success("保存成功");
        }
      } else {
        const result = await putPage(nowRoute.query.id, {
          title: titleConfig.value,
          blocks: [],
        });
        if (result) {
          message.success("保存成功");
        }
      }
      break;
    case "iframeReady":
      if (typeof nowRoute.query.id === "string") {
        const { title } = await getPageDetail(nowRoute.query.id);
        titleConfig.value = title;
      }
      break;
    default:
      break;
  }
};
const handleSave = () => {
  postMessageToIframe({
    type: "savePage",
  });
};
const handleNull = () => {
  message.warning("施工中");
};
useListener(handleMessage);
```

- 事件处理中心：响应来自两端的事件信号，分发事件。

```ts
import { onUnmounted } from "vue";
import { PostMessageType } from "./iframeType";
// 父通知子组件
export const postMessageToIframe = (payload: PostMessageType): void => {
  const iframe = document.querySelector("iframe");
  console.log("(tag)", payload);
  if (iframe) {
    iframe.contentWindow?.postMessage(payload, "*");
  }
};
//子组件通知父组件
export const postMessageToTop = (payload: PostMessageType): void => {
  if (window.top) {
    window.top.postMessage(payload, "*");
  }      You, 2 months ago • feat: new feature
};

// 监听消息hook
export const useListener = (
  handleMessage: (ev: MessageEvent<PostMessageType>) => unknown
): void => {
  window.addEventListener("message", handleMessage);
  onUnmounted(() => {
    window.removeEventListener("message", handleMessage);
  });
};
// 父组件的状态变化
export const asd = (): void => {
  const iframe = document.querySelector("iframe");
  console.log("tag", iframe);
};
```

3. 演示器分层，保证编辑时效果与实际表现的一致性，并可作为独立项目发布。
    - 展示层

      始终存在。

- 浮动层

  作为编辑器中的演示存在，在编辑器中响应点击事件（如编辑弹窗中数据等），并阻止事件传递到互动层。

- 互动层

  作为应用发布时存在，响应预设的产品交互事件，并阻止浮动层的事件响应。

4. 一键发布，即时预览编辑后效果

可靠性保证

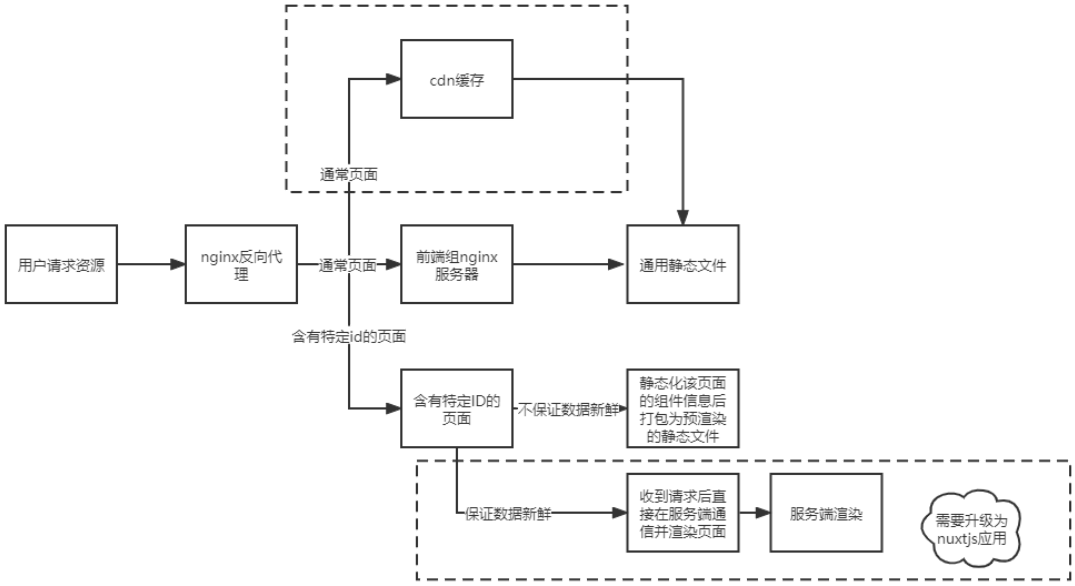
1. 缩小bug的影响范围，通过JSONschema约束组件接口传值缩小影响范围，保证数据符合前端规范后开始渲染，避免单个组件因数据问题导致整体应用出错，为后续提供单测用例打基础。
2. 接入移卡性能监控平台。

延展性

为应对可能出现的性能问题以及高并发场景，演示器（C端应用）的发布可以在不影响编辑器以及现有发布模式（B端应用）的情况下单独进行配置的渐进式升级：

1. 按需加载：编写合适的分包逻辑，为所有组件做单独打包为独立静态文件，请求页面信息后拉取对应该页面所需组件，减少单页面请求数据总量。
2. cdn缓存：打包为静态文件整体在公司cdn缓存，发布后更新cdn资源。
3. 服务端渲染：部分页面，如特定id的页面（比如某个应用的首页）等可能存在请求量过大的情况，可以为演示器升级为nuxtjs应用，为单独id的页面做预渲染打包处理，在服务端编写数据更新逻辑并返回相应的页面。

4. 上述方案都可以用nginx做首台反向代理的服务器，以是否含有特定页面id作为过滤条件，返回cdn/预渲染应用方案。



注：虚线部分为可升级部分