САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: “Фронт-энд разработка”

Отчет
Лабораторная работа №3

Выполнил: Митурский Богдан Антонович

Группа: K33392

Санкт-Петербург

2023 г.

Цель:

Доработать игру. Реализовать рутизацию в приложении, вынести общие компоненты в отдельную папку, реализовать работу страницы рейтинг:

Программное обеспечение: TypeScript, Node Typescript, Socket.io, React

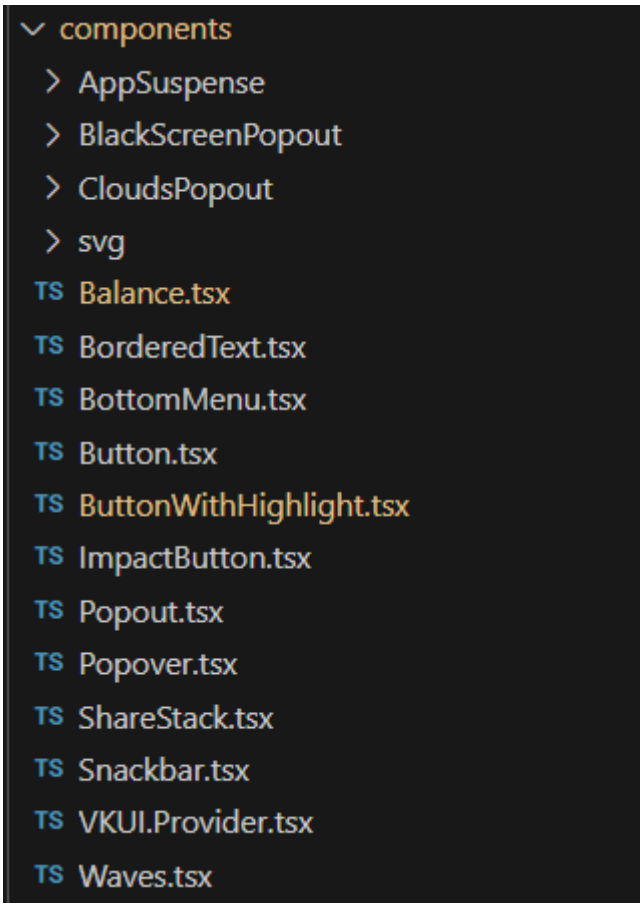
Ход работы:

1. Реализуем рутизацию в корневом компоненте используя библиотеку vkui из-за её анимаций и адаптированности под мини-приложения ВКонтакте:

```
<SplitLayout
  modal={<ModalController />}
  style={{ justifyContent: "center" }}
  popout={
    _id === 0 || isShopLoading || storyStatus === "loading" ? (
      <ScreenSpinner />
    ) : (
      popout
    )
  }
>
  <SplitCol animate={!isDesktop} width="100%" maxWidth="100%">
    <Epic id="/" tabbar={<BottomMenu />}>
      <View id="/">
        <LobbyPanel id="/lobby" mode="creator" />
        <ShopPanel id="/shop" />
        <ProfilePanel id="/profile" />
        <MainPanel id="/main" />
        <RatingPanel id="/rating" />
        <GamePanel id="/game" sprites={sprites} />
        <GameBotPanel id="/gameBot" sprites={sprites} />
      </View>
    </Epic>
  </SplitCol>
  <BlackScreenPopout>
    <EndGame id="endGame" />
    <StartGameTimer id="startGameTimer" />
  </BlackScreenPopout>
  <CloudsPopout>
    <WaitScreen id="waitScreen" />
    <LoaderProgress id="loaderProgress" />
    <GameResult id="gameResult" />
    <RedirectingClouds id="redirectingClouds" />
```

```
        </CloudsPopout>
        <ChestOpening />
        <Tutorial />
      </SplitLayout>
```

2. В качестве роутера будем использовать роутер от коммьюнити vkui - @iztevikat/router. Такое решение принимается для удобной работы с модальными окнами и алертами доступными в vkui.
3. Реализуем общую папку с компонентами и вынесем туда ключевые компоненты:



4. Добавим в компоненты базовую кнопку, использующуюся на разных страницах (ImapactButton):

```
import React from "react";
import styled from "@emotion/styled";
import BorderedText from "./BorderedText";
import { cx } from "@emotion/css";

const rootBackgroundColor = {
  primary: "#FFA401",
  secondary: "#4A70F4",
  negative: "#FC1E1E",
```

```
};
const rootBoxShadow = {
  primary:
    "0px 1px 0px #2B2B2B, inset 0px 3px 0px rgba(255, 255, 255, 0.12), inset
0px 2px 0px rgba(255, 255, 255, 0.36), inset 0px 3px 2px rgba(255, 255, 255,
0.24), inset 0px -6px 0px #B76200, inset 0px -6px 0px 1px rgba(255, 255, 255,
0.24);",
  secondary:
    "0px 1px 0px #2B2B2B, inset 0px 3px 0px rgba(255, 255, 255, 0.12), inset
0px 2px 0px rgba(255, 255, 255, 0.36), inset 0px 3px 2px rgba(255, 255, 255,
0.24), inset 0px -6px 0px #2F48A0, inset 0px -6px 0px 1px rgba(255, 255, 255,
0.24);",
  negative:
    "0px 1px 0px #212121, inset 0px 3px 0px rgba(255, 255, 255, 0.12), inset
0px 2px 0px rgba(255, 255, 255, 0.5), inset 0px -5px 0px rgba(0, 0, 0, 0.24)",
};
const contentBackgroundColor = {
  primary: "#FFB901",
  secondary: "#597EFF",
  negative: "linear-gradient(180deg, #FF3933 0%, #FF2A23 100%)",
};
const contentBoxShadow = {
  primary: "0px 2px 2px #FB8D00",
  secondary: "0px 2px 2px #3A60E8",
  negative: "",
};

type modeProps = {
  mode: "primary" | "secondary" | "negative";
  height?: number;
  width?: number;
  disabled?: boolean;
};
const Root = styled("button", { shouldForwardProp: (p) => p !== "mode" })(
  ({ mode }: modeProps) => ({
    backgroundColor: rootBackgroundColor[mode],
    borderRadius: 8,
    outline: "none",
    border: "1px solid #000000",
    boxShadow: rootBoxShadow[mode],
    padding: "4px 3px 10px 3px",
    display: "flex",
    flexDirection: "column",
    alignItems: "center",
    transitionDuration: "160ms",
    transitionTimingFunction: "ease-out",
    WebkitTapHighlightColor: "transparent",
    cursor: "pointer",
```

```
    userSelect: "none",
    "&:active": {
      transform: "scale(0.95)",
    },
  }),

  ({ height }: { height?: number }) => ({
    height: `${height}px`,
  }),

  ({ width }: { width?: number }) => ({
    width: `${width}px`,
  }),

  ({ disabled }: { disabled?: boolean }) => ({
    opacity: disabled ? 0.5 : 1,
    pointerEvents: disabled ? "none" : "auto",
  })
);

const Content = styled("div", { shouldForwardProp: (p) => p !== "mode" })(
  ({ mode }: modeProps) => ({
    background: contentBackgroundColor[mode],
    boxShadow: contentBoxShadow[mode],
    borderRadius: 6,
    flexGrow: 1,
    width: "100%",
    display: "flex",
    alignItems: "center",
    justifyContent: "center",
    flexDirection: "column",
    gap: 4,
    padding: "6px",
  })
);

const Title = styled(BorderedText, {
  shouldForwardProp: (p) => p !== "hasSubtitle",
})(({ hasSubtitle }) => ({
  fontFamily: "Russo One",
  fontSize: hasSubtitle ? 16 : 24,
  letterSpacing: "0.28px",
  paddingBottom: 2,
}));

const SmallTitle = styled(BorderedText)`
  font-family: Russo One;
  font-size: 12px;
```

```
  letter-spacing: 0.28px;
  padding-bottom: 2px;
`;

const Subtitle = styled("span")({
  fontSize: 11,
  color: "rgba(255, 255, 255, 0.75)",
  padding: "6px 0 4px 0",
});

type ImpactButtonProps = React.ComponentProps<"button"> & {
  children: React.ReactNode;
  mode?: "primary" | "secondary" | "negative";
  subtitle?: string;
  title?: string;
  rootIcon?: React.ReactElement;
  mainIcon?: string;
  secondIcon?: string;
  height?: number;
  width?: number;
  disabled?: boolean;
  className?: string;
  secondIconWidth?: number | string;
  mainIconWidth?: number;
};

const Icon = styled("img")``;

const Container = styled("div")<{ gap: string }>`
  display: flex;
  align-items: center;
  gap: ${(props) => props.gap};
`;

const ImpactButton = ({
  children,
  subtitle,
  mode = "primary",
  title,
  rootIcon,
  mainIcon,
  secondIcon,
  height = 80,
  width = 138,
  disabled = false,
  className,
  secondIconWidth,
  ...props
```

```tsx
}: ImpactButtonProps) => {
  const hasSubtitle = React.useMemo(() => !!subtitle, [subtitle]);
  const hasTitle = React.useMemo(() => !!title, [title]);

  return (
    <Root
      disabled={disabled}
      height={height}
      width={width}
      mode={mode}
      className={cx("ImpactButton", className)}
      {...props}
    >
      <Content mode={mode} className="ImpactButton__content">
        <Container gap="2px" className="ImpactButton__container">
          {rootIcon}
          {typeof children === "number" || typeof children === "string" ? (
            <Title
              hasSubtitle={hasSubtitle}
              className="ImpactButton__title"
              component="span"
              y={hasSubtitle ? 1 : 2}
              borderWidth={2.5}
              borderColor="rgba(0, 0, 0, 0.8)"
              shadowColor="rgba(0, 0, 0, 0.8)"
            >
              {children}
            </Title>
          ) : (
            children
          )}
          {hasTitle && <Icon style={{ width: "30px" }} src={mainIcon}></Icon>}
        </Container>
        {title && (
          <Container gap="4px">
            <SmallTitle>{title}</SmallTitle>
            <Icon width={secondIconWidth} src={secondIcon}></Icon>
          </Container>
        )}
      </Content>
      {hasSubtitle && (
        <Subtitle className="ImpactButton__subtitle">{subtitle}</Subtitle>
      )}
    </Root>
  );
};

export default React.memo(ImpactButton);
```

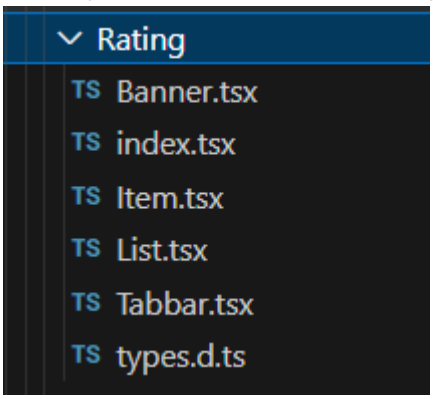
5. Также вынесем в компоненты счетчик баланса вместе с его логикой, т.к. он используется почти на всех страницах:

```
import React, { useEffect, useRef } from "react";
import styled from "@emotion/styled";
import { useSelector } from "react-redux";

import { icons } from "../data/url";

import { RootState } from "../store";
import BorderedText from "./BorderedText";
import anime from "animejs";
const Header = styled.div`
  /* margin: 14px 16px 0; */
  width: calc(100% - 16px);
  height: 34px;
  gap: 8px;
  display: flex;
  flex-direction: row;
  position: absolute;
  top: 14px;
  left: 16px;
  z-index: 1;
  margin-top: var(--safe-area-inset-top) !important;
`;
const BalanceWrapper = styled.span`
  display: flex;
  flex-direction: row;
  justify-content: center;
  align-items: center;
  padding: 4px 7px;
  > div {
    display: flex;
    gap: 4px;
  }

  background: linear-gradient(
    180deg,
    rgba(255, 255, 255, 0.9) 0%,
    rgba(255, 249, 228, 0.765) 100%
  );
  border: 1px solid rgba(0, 0, 0, 0.5);
  box-shadow: 0px 1px 0px rgba(0, 0, 0, 0.5),
    inset 0px -2px 0px rgba(0, 0, 0, 0.12),
    inset 0px 2px 0px rgba(255, 255, 255, 0.24);
  border-radius: 10px;
```

```
  color: white;
  font: 17px Russo One;
  letter-spacing: -0.28px;

  paint-order: stroke fill;
`;
const RatingWrapper = styled.span`
  display: flex;
  flex-direction: row;
  justify-content: center;
  align-items: center;
  gap: 4px;
  padding: 4px 7px;

  background: rgba(0, 0, 0, 0.3);
  border: 1px solid rgba(0, 0, 0, 0.5);
  box-shadow: 0px 1px 0px rgba(0, 0, 0, 0.5),
    inset 0px -2px 0px rgba(0, 0, 0, 0.12),
    inset 0px 2px 0px rgba(255, 255, 255, 0.24);
  border-radius: 10px;

  color: white;
  font: 17px Russo One;
  font-weight: 800;
  letter-spacing: -0.28px;

  paint-order: stroke fill;
`;
const Icon = styled.img`
  width: 20px;
  height: 20px;
`;

const Balance = () => {
  const { balance, rating } = useSelector((state: RootState) => state.User);
  const balanceRef = useRef<HTMLDivElement>(null);
  const prevBalanceRef = useRef<number>(balance);
  const handleClick = (e: React.MouseEvent<HTMLSpanElement>) => {
    e.preventDefault();
    e.stopPropagation();
  };

  useEffect(() => {
    if (
      prevBalanceRef.current !== undefined &&
      prevBalanceRef.current !== 0 &&
      balance > prevBalanceRef.current
```

```
    ) {
      anime({
        targets: balanceRef.current,
        keyframes: [
          { scale: 1.23, rotate: -1 },
          { scale: 0.93, rotate: 2 },
          { scale: 1, rotate: 0 },
        ],
        duration: 700,
        autoplay: true,
        easing: "cubicBezier(0.690, 0.545, 0.105, 1.460)",
      });
    }
    prevBalanceRef.current = balance;
  }, [balance]);

  return (
    <Header>
      <BalanceWrapper onClick={handleClick}>
        <div ref={balanceRef}>
          <Icon src={icons.coin} />
          <BorderedText>{balance}</BorderedText>
        </div>
      </BalanceWrapper>
      <RatingWrapper>
        <Icon src={icons.trophy} />
        <BorderedText>{rating}</BorderedText>
      </RatingWrapper>
    </Header>
  );
};

export default React.memo(Balance);
```

6. Реализуем страницу с отображением рейтинга игроков. По итогу она будет состоять из следующих блоков:



7. Реализуем работу с серверным api на уровне страницы рейтинга. Реализуем запрос общего рейтинга и рейтинга среди друзей, для которого, предварительно придется запросить список друзей у ВК, для отправки его на сервер друзей.

```
import React, { useState, useEffect } from "react";
import styled from "@emotion/styled";
import { Panel, Button, Placeholder } from "@vkontakte/vkui";

import Balance from "@/components/Balance";
import { request } from "@/utils/request";
import bridge from "@vkontakte/vk-bridge";
import { Icon56UserBookOutline, Icon56UsersOutline } from "@vkontakte/icons";

import Tabbar from "./Tabbar";
import Item from "./Item";
import List from "./List";
import Banner from "./Banner";

import "./types.d.ts";
import { images } from "@/data/url";
import { getAppId } from "@/utils/getAppId";

const Root = styled(Panel)({
  height: "100%",
  backgroundImage: `url(${images.background.rating})`,
  backgroundSize: "contain",
  "& > div": {
    height: "100%",
    display: "flex",
    flexDirection: "column",
    justifyContent: "center",
    alignItems: "center",
    padding: "0 8px",
    backgroundColor: "transparent !important",
  },
  "&:after": {
    backgroundColor: "transparent !important",
  },
});

const Card = styled("div")({
  height: "100%",
  overflowY: "auto",
  position: "relative",
  borderRadius: 14,
  background: "#EFF8FD",
```

```
  width: "100%",
  display: "flex",
  flexDirection: "column",
  boxShadow: "0 3px 0 0 rgba(255, 255, 255, 0.5)",
});

const ContentWrapper = styled.div`
  padding: 16px 8px;
  padding-top: 62px;
  width: 100%;
  height: 100%;
  display: flex;
  flex-direction: column;
  align-items: center;
  gap: 18px;
  max-width: 450px;
`;

const UserPlacement = styled.div`
  position: sticky;
  left: 0;
  right: 0;
  bottom: 0;
  background: #f8fcff;
  box-shadow: 0px -2px 0px rgba(0, 0, 0, 0.05);
  z-index: 1;
  padding: 8px;
`;

const RatingPanel: React.FC<{ id: string }> = () => {
  const [globalRating, setGlobalRating] = useState<IRatingState>();
  const [friendsRating, setFriendsRating] = useState<IRatingState>();
  const [season, setSeason] = useState<{ banner?: IBanner }>({});

  const [activeTabIndex, setActiveTabIndex] = useState(0);
  const rating = activeTabIndex === 0 ? globalRating : friendsRating;

  const appId = getAppId();

  const fillUsersFields = (items: ServerRatingItem[]) => {
    items.forEach((user: RatingItem) => {
      user.isStickersShow =
        activeTabIndex === 0 &&
        (season?.banner?.stickers || 0) + 1 > user.position;
    });
  };

  // Получение ключа пользователя для запроса списка друзей
```

```
const getAccessToken = async (
  scope: string
): Promise<
  | { access_token: string; error?: undefined }
  | { error: true; access_token?: undefined }
> => {
  try {
    const access_token = await bridge.send("VKWebAppGetAuthToken", {
      app_id: appId,
      scope,
    });
    return { access_token: access_token.access_token };
  } catch (err) {
    return { error: true };
  }
};

// Запрос рейтинга среди друзей
const getFriendsRating = async () => {
  setFriendsRating(undefined);

  const { access_token, error } = await getAccessToken("friends");
  if (error) return setFriendsRating({ error: "access_error_friends" });

  // Запрос списка друзей через ВК
  let { response: list } = await bridge.send("VKWebAppCallAPIMethod", {
    method: "friends.getAppUsers",
    params: { access_token, v: "5.131" },
  });
  list = list.join(",");

  const res = (await request({
    method: "user/friendsRating",
    auth: true,
    params: { list },
  })) as IResponse | { error: "no_friends" };

  setFriendsRating(res as IRatingState);
};

const fetchCurrentRating = async () => {
  if (activeTabIndex === 0) {
    const res = (await request({
      method: "user/rating",
      auth: true,
    })) as IResponse;

    fillUsersFields([...res.items, res.user]);
```

```
    setGlobalRating(res as IRatingState);
    return;
  }

  getFriendsRating();
};

useEffect(() => {
  (async () => {
    const seasonData = (await request({
      method: "user/season",
      auth: true,
    })) as { banner?: IBanner };
    if (seasonData.banner) seasonData.banner.endsAt += Date.now();

    setSeason(seasonData.banner ? seasonData : { banner: null });
  })();
}, []);

useEffect(() => {
  if (rating || (!season.banner && season.banner !== null)) return;

  fetchCurrentRating();
}, [season, activeTabIndex]);

return (
  <Root>
    <Balance />

    <ContentWrapper>
      {season.banner && <Banner data={season.banner} />}

      <Card>
        <Tabbar
          tabs={["Общий", "Друзья"]}
          activeTabIndex={activeTabIndex}
          setActiveTabIndex={setActiveTabIndex}
        />
        {!rating?.error && (
          <>
            <List data={rating?.items} season={season} />
            {rating?.user && (
              <UserPlacement>
                <Item data={rating?.user} />
              </UserPlacement>
            )}
          </>
        )}
```

```
          {rating?.error === "access_error_friends" && (
            <Placeholder
              icon={<Icon56UsersOutline />}
              header="Нет доступа"
              stretched
              action={
                <Button size="s" onClick={fetchCurrentRating}>
                  Разрешить
                </Button>
              }
            >
              Разрешите приложению
              <br />
              получить список ваших друзей
            </Placeholder>
          )}

          {rating?.error === "no_friends" && (
            <Placeholder icon={<Icon56UserBookOutline />} stretched>
              Ваших друзей пока нет в приложении,
              <br /> но вы можете пригласить их!
            </Placeholder>
          )}
        </Card>
      </ContentWrapper>
    </Root>
  );
};

export default React.memo(RatingPanel);
```

Вывод:

В ходе работы я реализовал страницу рейтинга, проработал структуру приложения, привязал страницу рейтинга к бекенду. В результате чего получилась вот такая страница: