7. pielikums

Programmas pirmkods.

File: **package.json**

```
{
    "name": "cardgame",
    "version": "0.0.1",
    "private": true,
    "scripts": {
        "dev": "pnpm i && concurrently \"vite dev\" \"npm:db\"
\"npm:tunnel\"",
        "build": "vite build",
        "preview": "concurrently \"vite preview\" \"npm:db\"",
        "db": "pocketbase serve",
        "tunnel": "ngrok http --domain=generous-adjusted-pegasus.ngrok-
free.app 8080",
        "check": "svelte-kit sync && svelte-check --tsconfig
./tsconfig.json",
        "check:watch": "svelte-kit sync && svelte-check --tsconfig
./tsconfig.json --watch",
        "lint": "prettier --plugin-search-dir . --check .",
        "format": "prettier --plugin-search-dir . --write ."
    },
    "devDependencies": {
        "@melt-ui/pp": "^0.3.2",
        "@melt-ui/svelte": "^0.81.0",
        "@skeletonlabs/skeleton": "^2.10.0",
        "@skeletonlabs/tw-plugin": "^0.4.0",
        "@sveltejs/adapter-auto": "^3.2.1",
        "@sveltejs/adapter-static": "^3.0.1",
        "@sveltejs/kit": "^2.5.10",
        "@sveltejs/vite-plugin-svelte": "^3.1.1",
        "@tailwindcss/forms": "^0.5.7",
        "@types/nanoid-dictionary": "^4.2.3",
        "@types/node": "^20.14.2",
        "@types/qrcode": "^1.5.5",
        "@zerodevx/svelte-toast": "^0.9.5",
        "autoprefixer": "^10.4.19",
        "concurrently": "^8.2.2",
        "postcss": "^8.4.38",
        "postcss-load-config": "^6.0.1",
        "prettier": "^3.3.1",
        "prettier-plugin-svelte": "^3.2.4",
        "svelte": "^4.2.18",
        "svelte-check": "^3.8.0",
        "svelte-sequential-preprocessor": "^2.0.1",
        "tailwindcss": "^3.4.4",
        "tslib": "^2.6.3",
        "typescript": "^5.4.5",
        "vite": "^5.2.13"
    },
    "type": "module",
    "dependencies": {
        "@csstools/normalize.css": "^12.1.1",
        "@faker-js/faker": "^8.4.1",
        "nanoid": "^5.0.7",
        "nanoid-dictionary": "^4.3.0",
        "pocketbase": "0.21.3",
        "qrcode": "^1.5.3",
        "sanitize.css": "^13.0.0",
        "sass": "^1.77.4"
    }
}
```

File: **src/app.postcss**

```
@tailwind base;
@tailwind components;
@tailwind utilities;

.tab-list {
    @apply print:hidden;
}

body {
    background: radial-gradient(circle at center, rgb(var(--color-surface-400)) 0%, #fffff0 100%);
    background-attachment: fixed, fixed;
    background-size: contain, cover;
    background-position: center, center;
    background-repeat: no-repeat, no-repeat;
}

@media print {
    table {
        border: 2px solid gray;
        width: 100vw !important;
    }
    td,
    th {
        border: 2px solid gray;
        white-space: normal !important;
        max-width: 32vw;
    }
}
```

File: **src/lib/components/DraggableObject.svelte**

```
<script lang="ts">
    import { topZIndex } from "$lib/draggableObjectHelper"
    import { createEventDispatcher } from "svelte"

    const emit = createEventDispatcher()

    let isPickedUp = false

    let thisZIndex = 0

    const mouseDown = (e: MouseEvent) => {
        // Uzstāda mainīgos
        const card = e.currentTarget as HTMLDivElement
        const rect = card.getBoundingClientRect()
        const offsetX = e.clientX - rect.left
        const offsetY = e.clientY - rect.top

        isPickedUp = true

        // Palielina globālo z-index lai pārējās kārtis sev uzliek
mazāku.
        // Šādā veidā z-index nebūs lielāks par 100 un kārtis nerādīsies
virs pop-up logiem.
        $topZIndex++
        thisZIndex = $topZIndex

        // Ja pele kustas, tad lai kārts tai seko.
        const onMouseMove = (e: MouseEvent) => {
            card.style.position = "absolute"
            card.style.left = `${e.clientX - offsetX}px`
            card.style.top = `${e.clientY - offsetY}px`
        }

        // Pelei paceļoties satīra aiz sevis
        const onMouseUp = (e: MouseEvent) => {
            window.removeEventListener("mousemove", onMouseMove)
            window.removeEventListener("mouseup", onMouseUp)

            isPickedUp = false

            // Ja kārts ir ārpus loga, to izņemt
            const rect = card.getBoundingClientRect()
            if (
                rect.left < 0 ||
                rect.right > window.innerWidth ||
                rect.top < 0 ||
                rect.bottom > window.innerHeight
            ) {
                emit("remove")
            }
        }

        window.addEventListener("mousemove", onMouseMove)
        window.addEventListener("mouseup", onMouseUp)
    }
</script>

<!-- svelte-ignore a11y-no-static-element-interactions -->
<div
    class="select-none transition-[transform] {isPickedUp && 'rotate-6
scale-110'}"
    style="z-index: {Math.max(20 - ($topZIndex - thisZIndex), 1)}"
    on:mousedown={mouseDown}
>
    <slot />
</div>
```

File: **src/lib/components/GameCard.svelte**

```
<script lang="ts">
    import { pb } from "$lib/database"
    import type { RecordModel } from "pocketbase"
    export let card: string | RecordModel

    import { getCard } from "$lib/cardCache"
</script>

{#if $pb}
    {#await typeof card == "string" ? getCard($pb, card) : card}
        <div class="card w-[13rem] h-[20rem] m-4 p-4 relative shadow-lg flex-shrink-0 rounded-lg" />
    {:then data}
        <div
            class="card w-[13rem] h-[20rem] m-4 p-3 relative shadow-lg flex-shrink-0 rounded-lg"
            style="background-color: color-mix(in hsl, {data?.custom?.color} 40%, rgb(var(--color-surface-100)))"
        >
            <h1 class="h3 font-cardtitle text-center pt-2">{data?.virsraksts}</h1>
            <p class="bg-surface-100 h-4/5 p-2 mt-2 rounded-lg">{data?.saturs}</p>

            <slot />
        </div>
    {/await}
{:else}
    <div class="card w-[13rem] h-[20rem] m-4 p-4 relative shadow-lg flex-shrink-0" />
{/if}
```

File: **src/lib/components/QrCode.svelte**

```
<script lang="ts">
    import QRCode from "qrcode"
    let display: HTMLCanvasElement

    export let url = "https://example.com"
    let dataUrl = ""

    $: QRCode.toDataURL(url, {}).then((url) => (dataUrl = url))
</script>

<img src={dataUrl} alt={url} class="rounded-lg" />
```

File: **src/routes/(home)/+page.svelte**

```
<script>
    import { account } from "$lib/account"
    import StartGame from "./StartGame.svelte"
</script>

<!--
    Sākumlapa
    Rāda dažādus elementus atkarībā no tā, vai lietotājs ir reģistrēts
-->

{#if !$account}
    <main
```

```
        class="flex flex-col items-center justify-center px-6 py-6 bg-
surface-100 mx-auto my-[40vh] w-max rounded-lg shadow-xl"
    >
        <h1 class="text-4xl font-bold">Projekts CardGame</h1>

        <p class="text-lg mt-6">Ienāc vai reģistrējies, lai sāktu savu
pirmo spēli!</p>
    </main>
{:else}
    <div class="w-full max-w-4xl p-6 mx-auto my-[25vh] bg-white rounded-
lg shadow-md">
        <h2 class="text-2xl font-semibold mb-4">Sāciet jaunu spēli</h2>
        <StartGame />
    </div>
{/if}

<!-- FAQ Section -->
<div class="w-full max-w-4xl p-6 mx-auto mt-10 card shadow-xl rounded-
lg">
    <h2 class="text-2xl font-semibold mb-4">Biežāk uzdotie jautājumi</h2>
    <div class="space-y-4">
        <div>
            <h3 class="text-lg font-medium">Kas šis ir?</h3>
            <p class="">
                CardGame ir tiešsaistes platforma, kas ļauj spēlēt
interaktīvas kartīšu spēles ar draugiem
                un kolēģiem.
            </p>
        </div>
        <div>
            <h3 class="text-lg font-medium">Kam ir CardGame domāts?</h3>
            <p class="">
                CardGame ir paredzēts ikvienam, kas vēlas izklaidēties un
veicināt komandas garu, izmantojot
                jautras un izglītojošas spēles.
            </p>
        </div>
        <div>
            <h3 class="text-lg font-medium">Kā es varu sākt?</h3>
            <p class="">
                Lai sāktu, vienkārši reģistrējies vai ienāc savā kontā,
izveido jaunu spēli un uzaicini
                draugus pievienoties.
            </p>
        </div>
    </div>
</div>

<div class="pb-10"></div>
```

File: **src/routes/(home)/StartGame.svelte**

```
<script lang="ts">
    import { account } from "$lib/account"
    import { pb } from "$lib/database"
    import { Stepper, Step } from "@skeletonlabs/skeleton"
    import type { RecordModel } from "pocketbase"
    let selectedCardSets: RecordModel[] = []

    /**
     * Funkcija tiek izsaukta, kad tiek pabeigts pēdējais solis.
     * Izveido spēli datubāzē un pāradresē uz to.
     */
    async function onCompleteHandler() {
        const data = {
            raditajs: $account?.id,
            secret: crypto.randomUUID(),
            noteikumi: gameRules,
            karsuKomplekti: selectedCardSets.map((x) => x.id)
        }

        const record = await $pb?.collection("speles").create(data)

        if (record) {
            console.log("record", record)
            location.href = `/game/host?id=${record.id}`
        }
    }

    let gameRules = {
        hostQuestionCards: 5,
        playerAnswerCards: 5,
        maxAnswersPerPlayer: 1,
        maxPlayers: 20
    }
</script>

<Stepper
    on:complete={onCompleteHandler}
    stepTerm="Solis"
    buttonNextLabel="Tālāk"
    buttonBackLabel="Atpakaļ"
    buttonCompleteLabel="Sākt spēli"
>
    <!--
        Solis 1
        Ievadsolis ar sākuma tekstu.
    -->
    <Step>
        <h1 class="h3">Izveidot jaunu spēli!</h1>
        <p>Izvēlies kāršu komplektus, dažus vienkāršus noteikumus, un sāc
spēli!</p>
    </Step>

    <!--
        Solis 2
        Izvēles solis, kurā tiek izvēlēti kāršu komplekti.
    -->
    <Step locked={selectedCardSets.length == 0}>
        <h1 class="h3">Izvēlies kāršu komplektus</h1>
        <p>Izvēlies kāršu komplektus, kuri tiks izmantoti spēlē.</p>

        <hr />

        <h1 class="h3 text-center">CardGame komplekti</h1>
        <div class="grid grid-cols-2 gap-4">
```

```
            <!--
                Ielādē visus oficiālos kāršu komplektus un tos parāda kā
izvēles pogas.
            -->
            {#await $pb
                ?.collection("karsuKomplekti")
                .getFullList({ filter: "official = true", requestKey:
"officialCardSets" })}
                <h1>Loading...</h1>
            {:then cardSets}
                {#each cardSets ?? [] as cardSet}
                    <button
                        class="card card-hover bg-surface-200
{selectedCardSets.includes(cardSet)
                            ? '!bg-success-200'
                            : ''}"
                        on:click={() => {
                            /*
                                Ja komplekts tiek izvēlēts, tas tiek
pievienots selectedCardSets masīvam
                            */
                            if (selectedCardSets.includes(cardSet)) {
                                selectedCardSets =
selectedCardSets.filter((x) => x !== cardSet)
                            } else {
                                selectedCardSets = [...selectedCardSets,
cardSet]
                            }
                        }}
                    >
                        <div class="card-header font-
bold">{cardSet.name}</div>
                        <div class="content">{cardSet.description}</div>
                    </button>
                {/each}
            {/await}
        </div>

        <div>
            <h1 class="h3 text-center">Mani komplekti</h1>

            <div class="grid grid-cols-2 gap-4">
                {#await $pb
                    ?.collection("karsuKomplekti")
                    .getFullList({ filter: `creator = "${$account?.id}"`,
requestKey: "myCardSets" })}
                    <h1>Ielādē...</h1>
                {:then cardSets}
                    {#each cardSets ?? [] as cardSet}
                        <button
                            class="card card-hover bg-surface-200
{selectedCardSets.includes(cardSet)
                                ? '!bg-success-200'
                                : ''}"
                            on:click={() => {
                                /*
                            Ja komplekts tiek izvēlēts, tas tiek
pievienots selectedCardSets masīvam
                        */
```

```
                                if (selectedCardSets.includes(cardSet)) {
                                    selectedCardSets =
selectedCardSets.filter((x) => x !== cardSet)
                                } else {
                                    selectedCardSets =
[...selectedCardSets, cardSet]
                                }
                            }}
                        >
                            <div class="card-header font-
bold">{cardSet.name}</div>
                            <div
class="content">{cardSet.description}</div>
                        </button>
                    {/each}
                {/await}
            </div>
        </div>
    </Step>

    <!--
        Solis 3
        Izvēles solis, kurā tiek izvēlēti noteikumi.
    -->
    <Step>
        <h1 class="h3">Izvēlies noteikumus</h1>
        <p>Izvēlies kādus noteikumus vēlies izmantot spēlē.</p>

        <div class="form grid grid-cols-3 gap-8">
            <label class="label">
                <span>Max. spēlēju skaits</span>
                <input type="number" class="input"
bind:value={gameRules.maxPlayers} min="0" max="20" />
            </label>

            <label class="label">
                <span>Vadītāja jautājuma kārtis</span>
                <input
                    type="number"
                    class="input"
                    bind:value={gameRules.hostQuestionCards}
                    min="2"
                    max="20"
                />
            </label>

            <label class="label">
                <span>Spēlēja atbilžu kārtis</span>
                <input
                    type="number"
                    class="input"
                    bind:value={gameRules.playerAnswerCards}
                    min="2"
                    max="20"
                />
            </label>

            <label class="label">
                <span>Max. atbilžu skaits</span>
                <input
                    type="number"
                    class="input"
                    bind:value={gameRules.maxAnswersPerPlayer}
                    min="1"
                    max="5"
                />
            </label>
        </div>
    </Step>
</Stepper>
```

File: **src/routes/+layout.svelte**

```
<script>
    import "../app.postcss"

    import LogInButton from "./LogInButton.svelte"
    import AccountButton from "./AccountButton.svelte"
    import { account } from "$lib/account"

    import { Accordion, AppBar, AppShell, Modal, Toast } from
"@skeletonlabs/skeleton"
    import { page } from "$app/stores"

    // Nepieciešams "toast" komponentam
    import { initializeStores } from "@skeletonlabs/skeleton"
    initializeStores()

    $: console.log(JSON.stringify($page.url.pathname.length))
</script>

<!--
    "Toast" ziņu pamatobjekts kas tiek izmantots lai parādītu ziņojumus
lietotājam.
-->
<Toast position="tr" />

<!--
    "Modal" komponents kas tiek izmantots lai parādītu modālu ziņas.
-->
<Modal />

<!--
    Galvenais lapas izkārtojums.
-->
<div>
    <div class="topbar print:hidden">
        {#if !$page.url.pathname.startsWith("/game/player")}
            <!--
            Lapas galvene
        -->
            <AppBar
                gridColumns="grid-cols-3"
                slotDefault="place-self-center"
                slotTrail="place-content-end"
                shadow="shadow-xl"
            >
```

```
                <!--
                Navigācijas pogas
            -->
                <svelte:fragment slot="lead">
                    <ul class="flex items-center justify-center space-x-
2">
                        <li>
                            <a
                                class="nav btn btn-sm
                                {$page.url.pathname === '/' ? 'variant-
outline-primary' : 'variant-filled-primary'}
                            "
                                href="/"
                            >
                                Sākums
                            </a>
                        </li>
                        <li>
                            <a
                                class="nav btn btn-sm
                                {$page.url.pathname === '/about' ?
'variant-outline-primary' : 'variant-filled-primary'}
                            "
                                href="/about"
                            >
                                Par mums
                            </a>
                        </li>
                    </ul>
                </svelte:fragment>

                <!--
                Logo
            -->
                <a class="home" href="/">
                    <div class="logo h3 font-bold">CardGame</div>
                </a>

                <!--
                Pārbaida vai lietotājs ir pierakstījies un parāda
atbilstošu pogu.
            -->
                <svelte:fragment slot="trail">
                    <ul>
                        {#if $account}
                            <AccountButton />
                        {:else}
                            <LogInButton />
                        {/if}
                    </ul>
                </svelte:fragment>
            </AppBar>
        {/if}
    </div>

    <slot />
</div>
```

File: **src/routes/AccountButton.svelte**

```
<script lang="ts">
    import { account } from "$lib/account"
    import { pb } from "$lib/database"
    import { createPopover, melt } from "@melt-ui/svelte"
    import { Avatar } from "@skeletonlabs/skeleton"

    const {
        elements: { trigger: pTrigger, content: pContent, arrow, close },
        states: { open }
    } = createPopover({
        forceVisible: true
    })
</script>

<ul>
    <li>
        {#if $account}
            <!--
                Lietotāja profila poga, kura atver lietotāja profila
Drop-Down izvēlni.
            -->
            <div class="account">
                <div class="name" use:melt={$pTrigger}>
                    <Avatar
                        initials={$account.name ?? $account.username}
                        border="border-2 border-surface-300-600-token
hover:!border-primary-500"
                        cursor="cursor-pointer"
                        width="w-10"
                    />
                </div>

                <!--
                    Drop-Down izvēlne ar lietotāja vārdu un izrakstīšanās
pogu.
                -->
                {#if $open}
                    <div class="card" use:melt={$pContent}>
                        <div class="card-header h5">
                            {$account.name ?? $account.username}
                        </div>
                        <div class="content p-3 flex flex-wrap flex-col
gap-1">
                            <a href="/user" class="btn variant-filled-
primary">Iestatījumi</a>
                            <!--
                                Izrakstīšanās poga. Tā izraksta lietotāju
un atjauno lapu.
                            -->
                            <button
                                class="btn variant-filled-error"
                                on:click={() => {
                                    $pb && $pb.authStore.clear()
                                    location.reload()
                                }}
                            >
                                Izrakstīties
                            </button>
                        </div>
                    </div>
                {/if}
            </div>
        {/if}
    </li>
</ul>
```

File: **src/routes/LogInButton.svelte**

```
<script>
    import { pb } from "$lib/database"
    import { createPopover, createTabs, melt } from "@melt-ui/svelte"
    import { Tab, TabGroup, getToastStore } from "@skeletonlabs/skeleton"

    import { fade, slide } from "svelte/transition"

    const toast = getToastStore()

    const tabs = [
        { id: "login", label: "Ienākt" },
        { id: "register", label: "Reģistrēties" }
    ]

    let currentTab = "login"

    const {
        elements: { trigger: pTrigger, content: pContent, arrow, close },
        states: { open }
    } = createPopover({
        forceVisible: true
    })

    const {
        elements: { root, list, content: tContent, trigger: tTrigger },
        states: { value }
    } = createTabs({
        defaultValue: "login"
    })

    /*
        Ievaddati tiek saglabāti lai pārslēdot uz reģistrācijas formu tie
nepazūd.
    */
    let input = {
        username: "",
        password: "",
        confirmPassword: ""
    }
</script>

<!--
    Ja lietotājs ir izrakstījies, tad tiek parādīta poga, kura atver
    pierakstīšanās un reģistrācijas Drop-Down izvēlni.
-->
<button use:melt={$pTrigger} type="button" class="btn variant-filled">
    Ienākt vai reģistrēties
```

```
</button>

<!--
    Drop-Down izvēlne ar pierakstīšanās un reģistrācijas formām.
-->
{#if $open}
    <div class="card" use:melt={$pContent}>
        <div class="card-header p-0">
            <!--
                Tabu grupa, kura izvēlas kuru no divām izvēlēm rādīt.
            -->
            <TabGroup>
                {#each tabs as tabItem}
                    <Tab bind:group={currentTab} name={tabItem.label}
value={tabItem.id}>
                        {tabItem.label}
                    </Tab>
                {/each}
            </TabGroup>
        </div>

        <div class="content p-3">
            {#if currentTab == "login"}
                <!--
                    Pierakstīšanās forma.
                -->
                <form
                    on:submit|preventDefault={(e) => {
                        /*
                        Pierakstās ar lietotājvārdu un paroli.
                        Ja tas izdodas, tad atjauno lapu.
                    */
                        $pb &&
                            $pb
                                .collection("lietotaji")
                                .authWithPassword(input.username,
input.password)
                                .then((x) => {
                                    location.reload()
                                })
                                .catch((err) => {
                                    console.log(err)
                                    toast.trigger({
                                        message: "Nepareizs lietotājvārds
vai parole!",
                                        background: "variant-filled-
warning"
                                    })
                                })
                    }}
                >
                    <label class="label">
                        <span>Lietotājvārds</span>
                        <input type="text" class="input"
bind:value={input.username} required />
                    </label>
                    <label class="label">
                        <span>Parole</span>
                        <input class="input" type="password"
bind:value={input.password} required />
                    </label>
                    <button type="submit" class="btn variant-filled block
mt-3 mx-auto"> Ienākt </button>
                </form>
            {:else if currentTab == "register"}
                <!--
                    Reģistrācijas forma.
                -->
                <form
                    on:submit|preventDefault={async (e) => {
                        /*
                                Pārbauda vai paroles sakrīt.
                            */
                        if (input.password != input.confirmPassword) {
                            toast.trigger({
                                message: "Paroles nesakrīt!",
                                background: "variant-filled-warning"
                            })
                            return
                        }

                        /*
                                Izveido lietotāju ar ievadīto
lietotājvārdu un paroli.
                                Ja tas izdodas, tad atjauno lapu.
                            */
                        $pb &&
                            $pb
                                .collection("lietotaji")
                                .create({
                                    name: input.username,
                                    email:
Math.random().toString(36).substring(2, 15) + "@example.com",
                                    username: input.username,
                                    password: input.password,
                                    passwordConfirm:
input.confirmPassword
                                })
                                .then((x) => {
                                    alert(
                                        "Reģistrācija izdevās! Lūdams
pierakstīties ar Jūsu lietotājvārdu un paroli."
                                    )
                                    location.reload()
                                })
                                .catch((err) => {
                                    console.log(err.data.data)
                                    toast.trigger({
                                        message: "Reģistrācija
neizdevās!",

                                        background: "variant-filled-

warning"
                                    })
                                })
                    }}
                >
                    <label class="label">
```

```
                        <span>Lietotājvārds</span>
                        <input type="text" class="input"
bind:value={input.username} required />
                    </label>
                    <label class="label">
                        <span>Parole</span>
                        <input type="password" class="input"
bind:value={input.password} required />
                    </label>
                    <label class="label">
                        <span>Parole (atkārtoti)</span>
                        <input type="password" class="input"
bind:value={input.confirmPassword} required />
                    </label>
                    <button type="submit" class="btn variant-filled block
mt-3 mx-auto">
                        Reģistrēties
                    </button>
                </form>
            {/if}
        </div>
    </div>
{/if}
```

File: **src/routes/about/+page.svelte**

```
<script>
    import { faker } from "@faker-js/faker"
</script>

<!--
    Par mums lapa, pagaidām viltus teksts
-->

<div class="card m-5 max-w-3xl mx-auto">
    <h1 class="h1 card-heder text-center">Par mums</h1>

    <div class="p-3">
        <p>
            {faker.lorem.paragraphs(10)}
        </p>
    </div>
</div>
```

File: **src/routes/admin/+layout.svelte**

```
<script lang="ts">
    import { adminPb } from "$lib/database"
    import Login from "./Login.svelte"
</script>

{#if $adminPb?.authStore.model}
    <slot />
{:else}
    <Login />
{/if}
```

File: **src/routes/admin/+page.svelte**

```
<script lang="ts">
    import { goto, invalidateAll } from "$app/navigation"
    import { page } from "$app/stores"
    import { adminPb } from "$lib/database"
    import { Tab, TabGroup, Table, tableMapperValues } from
"@skeletonlabs/skeleton"
    import type { RecordModel } from "pocketbase"
    import { fade, fly } from "svelte/transition"

    /*
        Tabu navigācijas mainīgie
        TabSet nosaka kuru tabu parādīt
        Filter nosaka kādu filtru lietot tabu saturam
    */
    let tabSet = parseInt($page.url.searchParams.get("tab") ?? "") || 0
    let filter = $page.url.searchParams.get("filter") as string |
undefined

    $: {
        /*
            Atjaunojam lapas URL ja mainīgie mainās, lai pēc lapas
atjaunošanas
            parādītu to pašu saturu.
        */
        $page.url.searchParams.set("tab", tabSet.toString())
        filter
            ? $page.url.searchParams.set("filter", filter.toString())
            : $page.url.searchParams.delete("filter")
        goto(`?${$page.url.searchParams.toString()}`, {
            replaceState: true
        })
        selectedItemId = undefined
    }

    const tabTitles = ["Lietotāji", "Kāršu komplekti", "Kārtis",
"Spēles", "Spēlētāji"]

    let selectedItemId: string | undefined
</script>

<!--
    Spēles administrācijas lapa
    Satur informāciju par lietoājiem, kārtīm, spēlēm, un citu saturu.
-->
<div class="flex justify-center align-items-center gap-5 pt-3 pb-2">
    <h1 class="h3">Administratora rīki - {tabTitles[tabSet]}</h1>

    <button
        class="btn btn-primary variant-filled-primary print:hidden"
        on:click={() => {
            $adminPb?.authStore.clear()
            location.reload()
        }}
    >
        Izrakstīties
    </button>
```

```
</div>

<!--
    Administrāciojas lapas navigācija
-->
<div class="mx-5 mb-5 bg-surface-100 rounded-lg shadow-xl print:shadow-
none print:m-0">
    <TabGroup>
        <Tab bind:group={tabSet} name="Lietotāji"
value={0}>Lietotāji</Tab>
        <Tab bind:group={tabSet} name="Kāršu komplekti" value={1}>Kāršu
komplekti</Tab>
        <Tab bind:group={tabSet} name="Kārtis" value={2}>Kārtis</Tab>
        <Tab bind:group={tabSet} name="Spēles" value={3}>Spēles</Tab>

        <svelte:fragment slot="panel">
            {#if filter}
                <button
                    class="btn btn-primary variant-filled-primary"
                    on:click={() => {
                        filter = undefined
                    }}
                >
                    Noņemt filtrus
                </button>
            {/if}

            {#if tabSet == 0}
                {#await $adminPb?.collection("lietotaji").getFullList({
filter, sort: "-created" })}
                    <p>Ielādē ... </p>
                {:then data}
                    <Table
                        interactive={true}
                        on:selected={(e) => {
                            selectedItemId = e.detail[0]
                        }}
                        source={{
                            head: ["Lietotājvārds", "Vārds", "Izveidots",
"Pēdējās izmaiņas"],
                            body: tableMapperValues(data ?? [],
["username", "name", "created", "updated"]),
                            meta: tableMapperValues(data ?? [], ["id"])
                        }}
                    />
                {/await}
            {:else if tabSet == 1}
                {#await
$adminPb?.collection("karsuKomplekti").getFullList({ filter, sort: "-
created" })}
                    <p>Ielādē ... </p>
                {:then data}
                    <Table
                        interactive={true}
                        on:selected={(e) => {
                            selectedItemId = e.detail[0]
                        }}
                        source={{
                            head: [
                                "ID",
                                "Nosaukums",
                                "Oficiāls",
                                "Izveidotājs",
                                "Apraksts",
                                "Izveidots",
                                "Pēdējās izmaiņas"
                            ],
                            body: tableMapperValues(data ?? [], [
                                "id",
                                "name",
                                "official",
                                "creator",
                                "description",
                                "created",
                                "updated"
                            ]),
                            meta: tableMapperValues(data ?? [], ["id"])
                        }}
                    />
                {/await}
            {:else if tabSet == 2}
                {#await
$adminPb?.collection("spelesKartis").getFullList({ filter, sort: "-
created" })}
                    <p>Ielādē...</p>
                {:then data}
                    <Table
                        interactive={true}
                        on:selected={(e) => {
                            selectedItemId = e.detail[0]
                        }}
                        source={{
                            head: ["ID", "Kāršu komplekts", "Tips",
"Virsraksts", "Saturs", "Izveidots"],
                            body: tableMapperValues(data ?? [], [
                                "id",
                                "karsuKomplekts",
                                "tips",
                                "virsraksts",
                                "saturs",
                                "created"
                            ]),
                            meta: tableMapperValues(data ?? [], ["id"])
                        }}
                    />
                {/await}
            {:else if tabSet == 3}
                {#await $adminPb?.collection("speles").getFullList({
filter, sort: "-created" })}
                    <p>Ielādē...</p>
                {:then data}
                    <Table
                        interactive={true}
                        on:selected={(e) => {
                            selectedItemId = e.detail[0]
                        }}
                        source={{
```

```
                            head: ["ID", "Radītājs", "Noslēpums",
"Noteikumi", "Komplekti", "Izveidots"],
                            body: tableMapperValues(data ?? [], [
                                "id",
                                "raditajs",
                                "secret",
                                "noteikumi",
                                "karsuKomplekti",
                                "created"
                            ]),
                            meta: tableMapperValues(data ?? [], ["id"])
                        }}
                    />
                {/await}
            {/if}
        </svelte:fragment>
    </TabGroup>
</div>

{#key selectedItemId}
    <div
        transition:fly={{ y: 100, duration: 200 }}
        class="absolute bottom-5 w-10/12 right-1/2 translate-x-1/2"
    >
        {#if selectedItemId}
            <div class="flex justify-center align-middle gap-5 p-3 card
shadow-lg">
                <button
                    class="btn btn-primary variant-filled-error"
                    on:click={() => {
                        $adminPb
                            ?.collection(["lietotaji", "karsuKomplekti",
"spelesKartis", "speles"][tabSet])
                            .delete(selectedItemId ?? "")
                        selectedItemId = undefined
                        location.reload()
                    }}
                >
                    Dzēst
                </button>
                <button
                    class="btn btn-primary variant-filled"
                    on:click={() => {
                        selectedItemId = undefined
                    }}
                >
                    Atcelt
                </button>

                <!--
                    Īpašas darbības atkarībā no tabu satura
                    piem. skatīt kārtis komplektā no kāršu komplekta
lapas
                -->
                {#if tabSet == 0}
                    <button
                        class="btn btn-primary variant-filled"
                        on:click={() => {
                            tabSet = 3
                            filter = `raditajs = "${selectedItemId}"`
                        }}
                    >
                        Skatīt spēles
                    </button>
                {:else if tabSet == 1}
                    <button
                        class="btn btn-primary variant-filled"
                        on:click={() => {
                            tabSet = 2
                            filter = `karsuKomplekts = "$
{selectedItemId}"`
                        }}
                    >
                        Skatīt kārtis komplektā
                    </button>
                {/if}
            </div>
        {/if}
    </div>
{/key}
```

File: **src/routes/admin/Login.svelte**

```
<script lang="ts">
    import { adminPb } from "$lib/database"
    import { getToastStore } from "@skeletonlabs/skeleton"

    console.log("adminPb", $adminPb?.authStore.model)

    const toast = getToastStore()
</script>

<!--
    Administratora pierakstīšanās lapa
    Administratoram ir pilna pieeja datubāzei
-->

<form
    class="card card-body card-flat max-w-lg p-3 mx-auto mt-4"
    on:submit|preventDefault={async (e) => {
        e.preventDefault()
        const data = new FormData(e.currentTarget)

        $adminPb?.admins
            .authWithPassword(
                (data.get("username") ?? "").toString(),
                (data.get("password") ?? "").toString()
            )
            .then((result) => {
                console.log("result", result)
                location.reload()
            })
            .catch((error) => {
                console.log("error", error)
                toast.trigger({
                    message: "Nepareizs lietotājvārds vai parole",
                    background: "variant-filled-error"
                })
            })
    }}
>
    <h3 class="h3 text-center">Administrators</h3>

    <label class="label">
        Lietotājvārds
        <input
            type="text"
            name="username"
            class="input input-bordered"
            placeholder="Lietotājvārds"
            required
        />
    </label>

    <label class="label">
        Parole
        <input
            type="password"
            name="password"
            class="input input-bordered"
            placeholder="Parole"
            required
        />
    </label>

    <button class="btn btn-primary variant-filled-primary mt-3 mx-auto
block">Ienākt</button>
</form>
```

File: **src/routes/game/+layout.svelte**

```
<div class="bg-surface-100/50 backdrop-blur-sm z-50 min-h-screen my-0 pt-
4">
    <slot />
</div>
```

File: **src/routes/game/host/+layout.svelte**

```
<script>
    import { account } from "$lib/account"
    import { pb } from "$lib/database"
</script>

{#if $account}
    <slot />
{:else}
    <p class="loadingMsg">Loading account...</p>
{/if}
```

File: **src/routes/game/host/+page.svelte**

```
<script lang="ts">
    import NewQuestionCard from "./NewQuestionCard.svelte"
    import { browser } from "$app/environment"
    import { page } from "$app/stores"
    import { pb } from "$lib/database"
    import type { RecordModel } from "pocketbase"
    import { derived, readable } from "svelte/store"
    import { createCurrentGameMovesStore, createGameMoveStore } from
"./gameMoves"
    import GameCard from "$lib/components/GameCard.svelte"
    import CardSelect from "./CardSelect.svelte"
    import QrCode from "$lib/components/QrCode.svelte"
    import DraggableObject from "$lib/components/DraggableObject.svelte"
    import { fly } from "svelte/transition"
    import { scale } from "svelte/transition"
    import { getToastStore } from "@skeletonlabs/skeleton"

    const toast = getToastStore()

    const gameId = $page.url.searchParams.get("id") as string

    if (!gameId && browser) {
        location.href = "/"
    }

    const gameRecord = derived(
        pb,
        ($pb, set) => {
            if (!$pb) return
            console.log("gameId", gameId)
            $pb
                .collection("speles")
                .getOne(gameId, {
                    expand: "jautajumaKarts"
                })
                .then((gameRecord) => {
                    console.log("gameRecord", gameRecord)
                    set(gameRecord)
                })

            $pb?.collection("speles").subscribe(gameId, function (e) {
                console.log(e)
                set(e.record)
            })
        },
        undefined as undefined | RecordModel
    )

    const gamePlayers = derived(
        pb,
        ($pb, set, update) => {
            if (!$pb) return

            $pb
                .collection("speletaji")
                .getFullList({
                    filter: `game = "${gameId}"`
                })
                .then((gamePlayers) => {
                    set(gamePlayers)
                    $pb?.collection("speletaji").subscribe("*", function
(e) {
                        if (e.record.game !== gameId) return

                        switch (e.action) {
                            case "create":
                                update((players) => [...players,
e.record])
                                break
                            case "update":
                                update((players) => {
                                    const index =
players.findIndex((player) => player.id === e.record.id)
                                    players[index] = e.record
                                    return players
                                })
                                break
                            case "delete":
                                update((players) => {
                                    const index =
players.findIndex((player) => player.id === e.record.id)
                                    players.splice(index, 1)
                                    return players
                                })
                                break
                        }
                    })
                })
        },
        [] as RecordModel[]
    )

    const gameMoves = createCurrentGameMovesStore(gameId, gamePlayers,
pb)

    let selectNewQuestionCard = false
</script>

<div class="gameBoard">
    {#if selectNewQuestionCard}
        <!--
    Izveido popup logu kurā tiek parādītas visas kārtis no izvēlētājiem
komplektiem.
-->

        <NewQuestionCard {gameRecord} {gameMoves}
bind:selectNewQuestionCard {gameId}></NewQuestionCard>
    {/if}

    {#if $gameRecord}
        {@const gameUrl = new URL(
            `/game/player?id=${gameId}&secret=${$gameRecord?.secret}`,
            $page.url.href
        ).href}
        <div class="qr card p-2 grid grid-rows-[max-content_1fr] h-max m-
2 justify-center text-center">
            <h3 class="font-bold text-lg w-full">Noskenē QR kodu lai
pievienotos!</h3>
```

```
            <a href={gameUrl} target="_blank" class="flex">
                <QrCode url={gameUrl} />
            </a>
        </div>

        <div class="players card h-max grid grid-cols-1 gap-1 p-3">
            <div class="h3 text-center">Spēlētāji:</div>
            {#each $gamePlayers ?? [] as player (player.id)}
                <!-- svelte-ignore a11y-click-events-have-key-events -->
                <!-- svelte-ignore a11y-no-static-element-interactions --
>
                <div
                    class="card p-1 text-center bg-surface-300 card-hover
cursor-pointer hover:bg-error-300"
                    transition:scale
                    on:click={() => {
                        // Delete player
                        $pb
                            ?.collection("speletaji")
                            .delete(player.id)
                            .then(() => {
                                $gamePlayers = $gamePlayers.filter((p) =>
p.id !== player.id)

                                toast.trigger({
                                    message: "Spēlētājs izdzēsts!",
                                    background: "variant-filled-error"
                                })
                            })
                    }}
                >
                    {player.name}
                </div>
            {/each}

            <a href="/game/host/moves?id={gameId}" class="btn variant-
filled-error mt-3">
                Beigt spēli!
            </a>
        </div>

        <hr />

        <div class="questionCard">
            {#if $gameMoves[0]}
                <GameCard card={$gameMoves[0].expand?.card}>
                    <button
                        class="btn variant-filled-primary absolute -
bottom-2 left-1/2 -translate-x-1/2"
                        on:click={() => {
                            $pb
                                ?.collection("spelesGajieni")
                                .create({
                                    player: undefined,
                                    game: gameId,
                                    card: null
                                })
                                .then(() => {
                                    selectNewQuestionCard = true
                                })
                        }}
                    >
                        Jauna kārts
                    </button>
                </GameCard>
            {:else}
                <button
                    class="btn variant-filled-primary"
                    on:click={() => {
                        selectNewQuestionCard = true
                    }}
                >
                    Jauna kārts
                </button>
            {/if}
        </div>

        <div class="flex flex-row flex-wrap answers">
            {#each $gameMoves.splice(1) ?? [] as move}
                <DraggableObject>
                    <GameCard card={move.card}>
                        <div
                            class="chip variant-filled m-1 absolute top-0
right-1/2 translate-x-1/2 -translate-y-4"
                        >
                            {move.expand?.player?.name}
                        </div>
                    </GameCard>
                </DraggableObject>
            {/each}
        </div>
    {/if}
</div>

<style>
    .gameBoard {
        display: grid;
        grid-template-columns: 1fr 1fr 15rem;
        grid-template-rows: min-content min-content 1fr;
        grid-column-gap: 0px;
        grid-row-gap: 0px;
        min-height: calc(100vh - 5rem);
    }

    .qr {
        grid-area: 1 / 3 / 3 / 4;
    }
    .questionCard {
        grid-area: 1 / 1 / 2 / 3;
    }
    .answers {
        grid-area: 2 / 1 / 4 / 3;
    }
    .players {
        grid-area: 3 / 3 / 4 / 4;
    }
</style>
```

File: **src/routes/game/host/CardSelect.svelte**

```
<script lang="ts">
    import GameCard from "$lib/components/GameCard.svelte"
    import { getToastStore } from "@skeletonlabs/skeleton"
    import type { RecordModel } from "pocketbase"
    import { writable } from "svelte/store"
    import { createGameMoveStore } from "../host/gameMoves"
    import { pb } from "$lib/database"
    import { random } from "nanoid"

    const toast = getToastStore()

    // Spēles kāršu komplekti
    export let gameCardSets: string[]
    // Spēles gājieni
    export let gameMoves: RecordModel[]
    // Vai rādīt šo izvēlni
    export let selectNewQuestionCard: boolean
    // Spēles ID
    export let gameId: string
    // Kāršu daudzums
    export let cardAmount: number

    console.log(gameCardSets)

    const cards = writable([] as RecordModel[], (set) => {
        $pb
            ?.collection("spelesKartis")
            .getFullList({
                filter: `tips = "jautajuma" && ( ${gameCardSets
                    .map((cardSet) => `karsuKomplekts = "${cardSet}"`)
                    .join(" || ")} )`
            })
            .then((cardSets) => {
                set(cardSets)
            })
    })
</script>

<div class="wrap">
    <div class="flex flex-wrap">
        {#each $cards.sort(() => Math.random() - 0.5).slice(0,
cardAmount) as card}
            <GameCard card={card.id}>
                <button
                    class="btn variant-filled-primary absolute -bottom-2
left-1/2 -translate-x-1/2"
                    on:click={() => {
                        $pb
                            ?.collection("spelesGajieni")
                            .create({
                                player: undefined,
                                game: gameId,
                                card: card.id
                            })
                            .then(() => {
                                toast.trigger({
                                    message: "Kārts izspēlēta!",
                                    background: "variant-filled-success"
                                })
                                selectNewQuestionCard = false
                            })
                    }}
                >
                    Izspēlēt
                </button>
            </GameCard>
        {:else}
            <p>Nav kāršu</p>
        {/each}
    </div>
</div>
```

File: **src/routes/game/host/NewQuestionCard.svelte**

```
<script lang="ts">
    import { browser } from "$app/environment"
    import { page } from "$app/stores"
    import { pb } from "$lib/database"
    import type { RecordModel } from "pocketbase"
    import { derived, readable, type Readable } from "svelte/store"
    import { createCurrentGameMovesStore, createGameMoveStore } from
"./gameMoves"
    import GameCard from "$lib/components/GameCard.svelte"
    import CardSelect from "./CardSelect.svelte"
    import { fade, fly, scale } from "svelte/transition"

    export let gameRecord: Readable<RecordModel>
    export let gameMoves: Readable<RecordModel[]>
    export let selectNewQuestionCard: boolean
    export let gameId: string
</script>

<div
    class="absolute inset-0 bg-surface-100 bg-opacity-75 z-50"
    style="display: block;"
    transition:scale={{ duration: 300 }}
>
    <div
        class="card absolute top-1/2 left-1/2 transform -translate-x-1/2
-translate-y-1/2 w-11/12 h-max shadow-xl"
    >
        <div class="modal-header">
            <h2 class="h2 text-center">Izvēlies jautājuma karti</h2>
        </div>
        <div class="modal-body">
            <div class="flex justify-center">
                <CardSelect
                    gameCardSets={$gameRecord?.karsuKomplekti}
                    gameMoves={$gameMoves}
                    bind:selectNewQuestionCard
                    {gameId}

cardAmount={$gameRecord?.noteikumi?.hostQuestionCards}
                />
            </div>
        </div>
    </div>
</div>
```

File: **src/routes/game/host/moves/+page.svelte**

```
<script lang="ts">
    import { page } from "$app/stores"
    import { derived } from "svelte/store"
    import { pb } from "$lib/database"
    import { createGameMoveStore } from "../gameMoves"
    import { Table, filter, type TableSource, tableMapperValues } from
"@skeletonlabs/skeleton"

    const gameId = $page.url.searchParams.get("id") as string

    if (!gameId) {
        location.href = "/"
    }

    const gameMoves = createGameMoveStore(gameId, pb)

    // Function to get the move details in a printable format
    const formatMove = (move: any) => {
        const playerName = move.expand?.player?.name ?? "Spēles vadītājs"
        const cardTitle = move.expand?.card?.virsraksts
        const cardContent = move.expand?.card?.saturs
        const time = new Date(move.created).toLocaleString()
        return {
            playerName,
            cardTitle,
            cardContent,
            time
        }
    }

    function createSource(data: any) {
        const tableSimple: TableSource = {
            // A list of heading labels.
            head: ["Laiks", "Vārds", "Kārts"],
            // The data visibly shown in your table body UI.
            body: data
                .filter((x: any) => x.card)
                .map((move: any) => {
                    return [
                        new Date(move.created).toLocaleString(),
                        move.expand?.player?.name,
                        `[${move.expand?.card?.virsraksts}] $
{move.expand?.card?.saturs}`
                    ]
                })
        }

        return tableSimple
    }
</script>

<div class="game-summary p-3 print:p-0">
```

```
    <h2 class="h1 text-center">Spēles gājieni</h2>

    <Table source={createSource($gameMoves)}></Table>

    <button
        on:click={() => {
            window.print()
        }}
        class="btn variant-filled-primary print:hidden"
    >
        Drukāt
    </button>
</div>
```

File: **src/routes/game/player/+layout.svelte**

```
<script>
    import { onMount } from "svelte"

    let name = ""

    onMount(() => {
        name = localStorage.getItem("player_name") ?? ""
    })
</script>

<!--
    Palūdz lietotāja vārdu pirms rāda UI
-->

{#if !name}
    <form
        on:submit|preventDefault={(e) => {
            let data = new FormData(e.currentTarget)
            console.log(data)
            let formName = data.get("name")?.toString() ?? ""
            console.log(formName)
            localStorage.setItem("player_name", formName)
            name = formName
        }}
        class="card max-w-96 p-3 mx-auto mt-20 text-center grid grid-
cols-1 gap-4"
    >
        <h1 class="h3 text-center">Vārds?</h1>
        <input type="text" name="name" id="name" class="input"
required />
        <button class="btn variant-filled-primary">Ienākt</button>
    </form>
{:else}
    <slot />
{/if}
```

File: **src/routes/game/player/+page.svelte**

```
<script lang="ts">
    import { browser } from "$app/environment"
    import { page } from "$app/stores"
    import { pb, playerPb } from "$lib/database"
    import { onMount } from "svelte"
    import { customAlphabet } from "nanoid"
    import CardSelect from "./CardSelect.svelte"
    import type { RecordModel } from "pocketbase"
    import { derived } from "svelte/store"
    import GameCard from "$lib/components/GameCard.svelte"
    import { createCurrentGameMovesStore, createGameMoveStore } from
"../host/gameMoves"

    const gameId = $page.url.searchParams.get("id") as string
    const secret = $page.url.searchParams.get("secret") as string

    if ((!gameId || !secret) && browser) {
        location.href = "/"
    }

    let playerRecord = $playerPb?.authStore.model
    console.log("playerRecord", playerRecord)

    /*
        Uzreiz kad spēlētājs pievienojas spēlei tam tiek prasīts vārds.
        Pēc tam programma mēģina izveidot spēlētāja ierakstu ar ID un
noslēpumu no URL.
        Tas neizdosies ja viens vai otrs ir nepareizi.
    */
    onMount(async () => {
        if (playerRecord && playerRecord.game == gameId) {
            return
        }

        // Izveido viltus lietotājvārdu un paroli
        // Tas ir nepieciešams lai izmantotu datu bāzes autentifikācijas
funkcijas
        const username =
customAlphabet("qwertyuiopasdfghjklzxcvbnm1234567890")()
        const password = crypto.randomUUID()
        const data = {
            username: username,
            email: crypto.randomUUID() + "@example.com",
            emailVisibility: true,
            password: password,
            passwordConfirm: password,
            game: gameId,
            name: localStorage.getItem("player_name") ?? "Spēlētājs",
            secret: secret
        }

        // Izveido lietotāju un izvēlas to kā aktīvo.
        const record = await
$playerPb?.collection("speletaji").create(data)
        const authData = await
$playerPb?.collection("speletaji").authWithPassword(username, password)

        //Saglabā lietotāju datus
        localStorage.setItem("player_user", username)
        localStorage.setItem("player_pass", password)

        location.reload()
    })

    /*
        Seko līdzi spēles ieraksta izmaiņām
    */
    const gameRecord = derived(
        playerPb,
        ($pb, set) ⇒ {
            if (!$pb) return
            console.log("gameId", gameId)
            $pb
                .collection("speles")
                .getOne(gameId)
                .then((gameRecord) ⇒ {
                    console.log("gameRecord", gameRecord)
                    set(gameRecord)
                })
                .catch((e) ⇒ {
                    // Ja ir notikusi kļūda, visdrīzāk spēlētājs ir
idzēsts
                    console.error(e)
                    if (localStorage.getItem("player_pass")) {
                        // Delete only entries with player_ prefix
                        Object.keys(localStorage).forEach((key) ⇒ {
                            if (String(key).startsWith("player_")) {
                                console.log("remove", key)
                                localStorage.removeItem(String(key))
                            }
                        })
                        location.reload()
                    }
                })

            $pb?.collection("speles").subscribe(gameId, function (e) {
                console.log(e)
                set(e.record)
            })
        },
        undefined as undefined | RecordModel
    )

    /*
        Seko līdzi jaunu spēlētāju pievienošanai
    */
    const gamePlayers = derived(
        playerPb,
        ($pb, set, update) ⇒ {
            if (!$pb) return
            $pb
                .collection("speletaji")
                .getFullList({
                    filter: `game = "${gameId}"`
                })
                .then((gamePlayers) ⇒ {
                    set(gamePlayers)
                })

            $pb?.collection("speletaji").subscribe("*", function (e) {
                if (e.record.game ≠ gameId) return
                switch (e.action) {
                    case "create":
                        update((players) => [...players, e.record])
                        break
                    case "update":
                        update((players) => {
                            const index = players.findIndex((player) =>
player.id === e.record.id)
                            players[index] = e.record
                            return players
                        })
                        break
                    case "delete":
                        update((players) => {
                            const index = players.findIndex((player) =>
player.id === e.record.id)
                            players.splice(index, 1)
                            return players
                        })
                        break
                }
            })
        },
        [] as RecordModel[]
    )

    /*
        Seko līdzi spēles gājieniem
    */
    let gameMoves = createCurrentGameMovesStore(gameId, gamePlayers,
playerPb)
</script>

{#if playerRecord && playerRecord.game === gameId}
    {#if $gameMoves[0]}
        <div class="w-max mx-auto">
            <GameCard card={$gameMoves[0].expand?.card} />
        </div>

        {#if $gameRecord?.karsuKomplekti}
            {#if $gameMoves.every((move) => move.player !==
playerRecord?.id)}
                <CardSelect
                    gameCardSets={$gameRecord?.karsuKomplekti}
                    gameMoves={$gameMoves}
                    maxCards={$gameRecord?.noteikumi?.playerAnswerCards}
                />
            {:else}
                <div class="w-max mx-auto card p-5 mt-10">Spēles kārts ir
izspēlēta!</div>
            {/if}
        {/if}
    {:else}
        <div class="grid justify-items-center content-center h-full p-4">
            <div class="card w-max p-4 font-bold">Spēles vadītājs izvēlas
kārti...</div>
        </div>
    {/if}
```

```
{/if}
```

File: **src/routes/game/player/CardSelect.svelte**

```
<script lang="ts">
    import GameCard from "$lib/components/GameCard.svelte"
    import { playerPb } from "$lib/database"
    import { getToastStore } from "@skeletonlabs/skeleton"
    import type { RecordModel } from "pocketbase"
    import { writable } from "svelte/store"
    import { createGameMoveStore } from "../host/gameMoves"
    import AccountButton from "../../AccountButton.svelte"

    const toast = getToastStore()

    export let gameCardSets: string[]
    export let gameMoves: RecordModel[]
    export let maxCards: number

    console.log(gameCardSets)

    /*
        Atrod visas atbilžu kārtis kuras var izspēlēt
    */
    const cards = writable([] as RecordModel[], (set) => {
        $playerPb
            ?.collection("spelesKartis")
            .getFullList({
                filter: `tips = "atbilzu" && ( ${gameCardSets
                    .map((cardSet) => `karsuKomplekts = "${cardSet}"`)
                    .join(" || ")} )`
            })
            .then((cardSets) => {
                set(cardSets)
            })
    })

    // Samaisa kārtis, garantējos ka vismaz viena kārts no katras
kategorijas
    // tiks parādīta
    function shuffleCards(cards: RecordModel[], maxCards: number):
RecordModel[] {
        // Ja neviena kārts neatbilst, atgriez tukšu
        if (!cards.length) return []

        // Izveido masīvu ar visiem virsrakstiem
        let allTitles = Object.keys(
            cards.reduce((acc: Record<string, boolean>, card) => {
                console.log(card.virsraksts)
                // Viegls veids kā tikt vaļā no duplikātiem
                acc[card.virsraksts as string] = true
                return acc
            }, {})
            // Uzreiz samaisa virsrakstus, lai pat ja ir vairāk
virsrakstu nekā kārts
            // visiem ir iespēja parādīties
        ).sort(() => Math.random() - 0.5)
        console.log({ allTitles })

        // Sadala pašas kārtis un tās samaisa
        let cardsPerTitle: Record<string, RecordModel[]> = {}
        for (const title of allTitles) {
            cardsPerTitle[title] = cards
                .filter((card) ⇒ card.virsraksts == title)
                .sort(() ⇒ Math.random() - 0.5)
        }

        console.log({ cardsPerTitle })

        // Izvelk vienu kārti no katras virsrakstu kaudzes līdz ir
sasniegts maxCards
        let finalCards: RecordModel[] = []
        for (let i = 0; finalCards.length < maxCards; i++) {
            const title = allTitles[i % allTitles.length]
            console.log({ title })

            // Ja kaudze tukša, mēģināt nākamo
            // TODO: Ja visas kaudzes beidzas, šis būs bezgalīgs
            if (!cardsPerTitle[title].length) {
                continue
            }

            // Izvelk kārti un pievieno gala masīvam
            const card = cardsPerTitle[title].shift() as RecordModel
            finalCards.push(card)
        }

        return finalCards
    }
</script>

<div class="wrap">
    <div class="flex flex-nowrap overflow-x-scroll snap-x snap-
mandatory">
        {maxCards}
        {#each shuffleCards($cards ?? [], maxCards) as card}
            <div class="flex-shrink-0 snap-center">
                <GameCard card={card.id}>
                    <button
                        class="btn variant-filled-primary absolute -
bottom-2 left-1/2 -translate-x-1/2"
                        on:click={() ⇒ {
                            /*
                                Izspēlē izvēlēto kārti
                            */
                            $playerPb
                                ?.collection("spelesGajieni")
                                .create({
                                    player:
$playerPb?.authStore.model?.id,
                                    game:
$playerPb?.authStore.model?.game,
                                    card: card.id
                                })
                                .then(() ⇒ {
                                    toast.trigger({
```

```
                    message: "Kārts izspēlēta!",
                    background: "variant-filled-
success"
                  })
                })
            }}
          >
            Izspēlēt
          </button>
        </GameCard>
      </div>
    {:else}
      <p>Nav kāršu</p>
    {/each}
  </div>
</div>
```

File: **src/routes/user/+layout.svelte**

```
<script>
  import { account } from "$lib/account"
  import { pb } from "$lib/database"
</script>

{#if $account}
  <slot />
{:else}
  <p class="loadingMsg">Loading account...</p>
{/if}
```

File: **src/routes/user/+page.svelte**

```
<script lang="ts">
  import { goto } from "$app/navigation"
  import { page } from "$app/stores"
  import { account } from "$lib/account"
  import { pb } from "$lib/database"
  import { TabGroup, Tab, TabAnchor } from "@skeletonlabs/skeleton"
  import CardSets from "./CardSets.svelte"
  import MyGames from "./MyGames.svelte"

  let tabSet = parseInt($page.url.searchParams.get("tab") ?? "") || 0
  $: {
    $page.url.searchParams.set("tab", tabSet.toString())
    goto(`?${$page.url.searchParams.toString()}`, {
      replaceState: true
    })
  }
</script>

<!--
  Lietotāja profila lapa, var redzēt savus datus un veikt izmaiņas.
-->

<div class="card max-w-5xl mx-auto mt-4 variant-glass">
  <TabGroup>
```

```
        <Tab bind:group={tabSet} name="Iestatījumi"
value={0}>Iestatījumi</Tab>
        <Tab bind:group={tabSet} name="Drošība" value={1}>Drošība</Tab>
        <Tab bind:group={tabSet} name="Komplekti" value={2}>Kāršu
komplekti</Tab>
        <Tab bind:group={tabSet} name="Vesture" value={3}>Spēļu
vēsture</Tab>

        <div slot="panel" class="p-4">
            {#if tabSet == 0}
                <form
                    on:submit|preventDefault={async (e) => {
                        e.preventDefault()
                        const form = e.currentTarget
                        const formData = new FormData(form)
                        const data =
Object.fromEntries(formData.entries())
                        $pb && (await
$pb.collection("lietotaji").update($account?.id ?? "", data))
                        location.reload()
                    }}
                >
                    <div>
                        <div>
                            <label class="label">
                                Vārds
                                <input type="text" name="name"
class="input" value={$account?.name} required />
                            </label>
                        </div>

                        <div>
                            <label class="label">
                                E-pasts
                                <input type="email" name="email"
class="input" value={$account?.email} required />
                            </label>
                        </div>
                    </div>

                    <button type="submit" class="btn btn-lg variant-
filled-primary mx-auto block mt-3">
                        Saglabāt
                    </button>
                </form>
            {:else if tabSet == 1}
                <!-- Mainīt paroli -->
                <form
                    on:submit|preventDefault={async (e) => {
                        e.preventDefault()
                        const form = e.currentTarget
                        const formData = new FormData(form)
                        const data =
Object.fromEntries(formData.entries())
                        $pb && (await
$pb.collection("lietotaji").update($account?.id ?? "", data))
                        location.reload()
                    }}
                >
```

```
                    <div>
                        <div>
                            <label class="label">
                                Parole
                                <input type="password" name="oldPassword"
class="input" required />
                            </label>
                        </div>

                        <div>
                            <label class="label">
                                Jaunā parole
                                <input type="password" name="password"
class="input" required />
                            </label>
                        </div>

                        <div>
                            <label class="label">
                                Jaunā parole atkārtoti
                                <input type="password"
name="passwordConfirm" class="input" required />
                            </label>
                        </div>
                    </div>

                    <button type="submit" class="btn btn-lg variant-
filled-primary mx-auto block mt-3">
                        Saglabāt
                    </button>
                </form>
            {:else if tabSet == 2}
                <!-- Kāršu komplekti -->
                <CardSets />
            {:else if tabSet == 3}
                <!-- Spēļu vēsture -->
                <MyGames />
            {/if}
        </div>
    </TabGroup>
</div>
```

File: **src/routes/user/CardSets.svelte**

```
<script lang="ts">
    import { account } from "$lib/account"
    import { pb } from "$lib/database"
    import { readable, writable } from "svelte/store"
    import type { RecordModel } from "pocketbase"

    // Izveido un pārvalda kāršu komplektus

    const myCardSets = writable([] as RecordModel[], (set) => {
        // Ielādē lietotāja izveidotos kāršu komplektus
        $pb
            ?.collection("karsuKomplekti")
            .getFullList({ filter: `creator = "${$account?.id}"` })
            .then((cardSets) => {
                set(cardSets)
            })
    })

    async function createCardSet(name: string, description: string) {
        // Izveido jaunu kāršu komplektu
        const cardSet = await $pb
            ?.collection("karsuKomplekti")
            .create({
                name,
                description,
                creator: $account?.id
            })
            .then((newCardSet) => {
                // Atjauno kāršu komplektu sarakstu
                myCardSets.update((cardSets) => [...cardSets,
newCardSet])
            })

        return cardSet
    }
</script>

<div class="wrap">
    <h1 class="h1">Mani kāršu komplekti</h1>

    <!--
        Visi kāršu komplekti
    -->
    {#each $myCardSets as cardSet}
        <div class="card my-3 p-3">
            <span class="font-bold">{cardSet.name} </span>
            <span class="content">{cardSet.description}</span>

            <div>
                <a class="btn btn-sm variant-filled-primary"
href="/user/cards?cardSet={cardSet.id}">
                    Rediģēt
                </a>
                <button
                    class="btn btn-sm variant-filled-error"
                    on:click={() => {
                        $pb
                            ?.collection("karsuKomplekti")
                            .delete(cardSet.id)
                            .then(() => {
                                myCardSets.update((cardSets) =>
cardSets.filter((c) => c.id !== cardSet.id))
                            })
                    }}
                >
                    Dzēst
                </button>
            </div>
        </div>
    {/each}

    <!--
        Jauna komplekta izveide
    →
    <div class="card p-3">
        <div class="card-header font-bold">Izveidot jaunu kāršu
komplektu</div>
        <div class="content">
            <form
                on:submit|preventDefault={async (e) => {
                    e.preventDefault()
                    const form = e.currentTarget
                    const formData = new FormData(form)
                    const data = Object.fromEntries(formData.entries())
                    await createCardSet(data.name.toString(),
data.description.toString()).then(() => {
                        form.reset()
                    })
                }}
            >
                <div>
                    <label class="label">
                        Nosaukums
                        <input type="text" name="name" class="input"
required />
                    </label>
                </div>

                <div>
                    <label class="label">
                        Apraksts
                        <textarea name="description" class="input"
required />
                    </label>
                </div>

                <button type="submit" class="btn btn-lg variant-filled-
primary mx-auto block mt-3">
                    Izveidot
                </button>
            </form>
        </div>
    </div>
</div>
```

File: **src/routes/user/MyGames.svelte**

```
<script lang="ts">
    import { account } from "$lib/account"
    import { pb } from "$lib/database"
    import type { RecordModel } from "pocketbase"
    import { writable } from "svelte/store"

    const myGames = writable([] as RecordModel[], (set) => {
        $pb
            ?.collection("speles")
            .getFullList({
                filter: `raditajs = "${$account?.id}"`,
                expand: "karsuKomplekti",
                sort: "-created"
            })
            .then((games) => {
                console.log("games", games)
                set(games)
            })
    })
</script>

<!--
    Spēļu vēsture
-->

<button
    on:click={() => {
        window.print()
    }}
    class="btn variant-filled-primary print:hidden"
>
    Drukāt
</button>

<div class="wrap max-w-3xl mx-auto printWrap">
    <h1 class="h3 text-center">Manas spēles</h1>

    {#each $myGames as game}
        <div class="card my-3 p-3 break-inside-avoid print:border-2
print:border-gray-800">
            <span class="font-bold">{game.id} </span>
            <br />
            <span class="text-surface-500">{new
Date(game.created).toLocaleString()}</span>
            <span class="content">
                <ul>
                    <li>
                        Spēles noteikumi: {#each
Object.entries(game.noteikumi ?? {}) as [noteikums, value]}
                            <span class="chip variant-filled-tertiary m-1
print:border-2 print:border-gray-800">
                                {noteikums}: {value}
                            </span>
                        {/each}
                    </li>
                    <li>
                        Kāršu komplekti: {#each
game.expand?.karsuKomplekti.map((x) => x.name) ?? [] as cardSetName}
                            <span class="chip variant-filled-secondary m-
1 print:border-2 print:border-gray-800">
                                {cardSetName}
                            </span>
                        {/each}
                    </li>
                    <a
                        href="/game/host/moves?id={game.id}"
                        class="btn btn-sm variant-filled-primary
print:hidden"
                    >
                        Skatīt spēles gājienus
                    </a>
                </ul>
            </span>
        </div>
    {:else}
        <div class="card my-3 p-3">
            <span class="font-bold">Nav spēļu</span>
        </div>
    {/each}
</div>
```

File: **src/routes/user/cards/+page.svelte**

```
<script lang="ts">
    import { page } from "$app/stores"
    import { pb } from "$lib/database"
    import type { RecordModel } from "pocketbase"
    import { writable } from "svelte/store"
    import { faker } from "@faker-js/faker"
    import { goto } from "$app/navigation"

    const cards = writable([] as RecordModel[], (set) => {
        $pb
            ?.collection("spelesKartis")
            .getFullList({
                filter: `karsuKomplekts = "$
{$page.url.searchParams.get("cardSet")}"`
            })
            .then((cardSets) => {
                set(cardSets)
            })
    })

    const cardSet = writable({} as RecordModel, (set) => {
        $pb
            ?.collection("karsuKomplekti")
            .getOne($page.url.searchParams.get("cardSet") ?? "")
            .then((cardSet) => {
                set(cardSet)
            })
    })

    async function generateExampleCards() {
        for (let i = 0; i < 20; i++) {
            const tips = i % 2 === 0 ? "jautajuma" : "atbilzu"

            const card = {
                virsraksts: `Kārts ${tips} ${i}`,
                saturs: faker.lorem.sentence(),
                karsuKomplekts: $page.url.searchParams.get("cardSet"),
                tips: tips,
                custom: "{}"
            }
            await $pb?.collection("spelesKartis").create(card)
        }
    }
</script>

<!--
    Kāršu komplekta rediģēšana
-->
```

```
<div class="card mx-5 my-3 p-3">
    <div class="font-bold text-center">Kāršu komplekts:
{$cardSet.name}</div>
    <div class="content">
        <form
            on:submit|preventDefault={async (e) => {
                e.preventDefault()
                const form = e.currentTarget
                const formData = new FormData(form)
                const data = Object.fromEntries(formData.entries())
                $pb && (await
$pb.collection("karsuKomplekti").update($cardSet.id ?? "", data))
                goto("/user?tab=2")
            }}
        >
            <label class="label">
                Nosaukums
                <input class="input" type="text" name="name" id="name"
value={$cardSet.name} required />
            </label>
            <label class="label">
                Apraksts
                <textarea class="input" name="description"
id="description" required
                    >{$cardSet.description}</textarea
                >
            </label>
            <button class="btn variant-filled-primary">Saglabāt</button>
        </form>
    </div>
</div>

<div class="max-w-2xl mx-auto">
    {#each $cards as card}
        <div class="card p-3 my-3 relative">
            <div class="font-bold">
                {card.virsraksts}
            </div>
            <div class="content">
                {card.saturs}
            </div>
            <button
                on:click={() => {
                    $pb
                        ?.collection("spelesKartis")
                        .delete(card.id)
                        .then(() => {
                            cards.update((cards) => cards.filter((c) =>
c.id !== card.id))
                        })
                }}
                class="btn btn-sm variant-outline-error absolute top-0
right-0"
            >
                Dzēst
            </button>
        </div>
    {:else}
```

```
        <div class="card p-3 my-3">
            <div class="content text-center">Komplektā kāršu nav!</div>
            <button
                on:click={async () => {
                    await generateExampleCards()
                    location.reload()
                }}
                class="btn variant-filled-primary block mx-auto mt-3"
            >
                Izveidot piemēra kārtis
            </button>
        </div>
    {/each}
</div>

<div class="card max-w-xl mx-auto p-3 mb-12">
    <div class="font-bold text-center">Izveidot jaunas kārtis</div>
    <div class="content">
        <form
            on:submit|preventDefault={async (e) => {
                e.preventDefault()
                const form = e.currentTarget
                const formData = new FormData(form)

                const allCards =
formData.get("saturi")?.toString().trim().split("\n")

                if (!allCards?.length) {
                    throw new Error("Kārtis nav definētas!")
                }

                for (const card of allCards) {
                    await $pb
                        ?.collection("spelesKartis")
                        .create({
                            virsraksts:
formData.get("title")?.toString(),
                            saturs: card,
                            karsuKomplekts:
$page.url.searchParams.get("cardSet"),
                            tips: formData.get("tips")?.toString() ??
"jautajuma",
                            custom:
JSON.parse(formData.get("custom")?.toString() ?? "{}")
                        })
                        .then((card) => {
                            cards.update((cards) => [ ... cards, card])
                            form.reset()
                        })
                }
            }}
        >
            <label class="label">
                Virsraksts
                <input class="input" type="text" name="title" id="title"
required />
            </label>
            <label class="label">
                Saturs (katra kārts jaunā rindā)
                <textarea class="input" name="saturi" id="saturi"
required></textarea>
            </label>
            <label class="label">
                Tips
                <select class="input" name="tips" id="tips" required>
                    <option value="jautajuma">Jautājuma</option>
                    <option value="atbilzu">Atbildes</option>
                </select>
            </label>
            <label class="label">
                Papildus dati (piem. krāsa, īpašas darbības, utt.)
                <textarea class="input" name="custom" id="custom"
required></textarea>
            </label>
            <button class="btn variant-filled-primary block mx-auto mt-
3">Izveidot</button>
        </form>
    </div>
</div>

<!-- Fix cards with "custom" as string instead of JSON -->
<button
    on:click={() => {
        let toUpdate = $cards.length
        $cards.forEach((card, i) => {
            if (typeof card.custom == "string") {
                console.log("fix", card.custom)

                setTimeout(async () => {
                    await $pb?.collection("spelesKartis").update(card.id,
{
                        custom: eval(`let i = ${card.custom}; i`)
                    })
                    toUpdate--
                    console.log(toUpdate)
                }, 100 * i)
            }
        })
    }}
    class="btn variant-filled-primary block mx-auto my-5"
>
    Salabot datu struktūru
</button>
```

File: **tailwind.config.ts**

```
import { join } from "path"
import type { Config } from "tailwindcss"
import forms from "@tailwindcss/forms"

// 1. Import the Skeleton plugin
import { skeleton } from "@skeletonlabs/tw-plugin"

const config = {
    // 2. Opt for dark mode to be handled via the class method
    darkMode: "class",
    content: [
        "./src/**/*.{html,js,svelte,ts}",
        // 3. Append the path to the Skeleton package
        join(require.resolve("@skeletonlabs/skeleton"), "../**/*.
{html,js,svelte,ts}")
    ],
    theme: {
        extend: {
            fontFamily: {
                cardtitle: ["'Poetsen One', sans-serif"]
            }
        }
    },
    plugins: [
        // 4. Append the Skeleton plugin (after other plugins)
        skeleton({
            themes: {
                preset: ["gold-nouveau"]
            }
        }),
        forms
    ]
} satisfies Config

export default config
```

File: **tsconfig.json**

```
{
    "extends": "./.svelte-kit/tsconfig.json",
    "compilerOptions": {
        "allowJs": true,
        "checkJs": true,
        "esModuleInterop": true,
        "forceConsistentCasingInFileNames": true,
        "resolveJsonModule": true,
        "skipLibCheck": true,
        "sourceMap": true,
        "strict": true
    }
    // Path aliases are handled by
https://kit.svelte.dev/docs/configuration#alias
    //
    // If you want to overwrite includes/excludes, make sure to copy over
the relevant includes/excludes
    // from the referenced tsconfig.json - TypeScript does not merge them
in
}
```

File: **vite.config.ts**

```
import { sveltekit } from "@sveltejs/kit/vite"
import { defineConfig } from "vite"

export default defineConfig({
    server: {
        port: 8080
    },
    plugins: [sveltekit()],
    envPrefix: [
        "CODESPACES",
        "GITHUB_CODESPACES_PORT_FORWARDING_DOMAIN",
        "CODESPACE_NAME"
    ]
})
```