**Факультет інформатики та обчислювальної техніки**

**Кафедра інформатики та програмної інженерії**

“ЗАТВЕРДЖЕНО”

\_\_\_\_\_\_\_\_\_\_\_\_ Максим Головченко

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 р.

**Програмне забезпечення для Веб-гра «Сапер» з можливістю авторизації та перегляду результатів ігор**

**Опис програми**

КПІ.ІП-9112.045440.05.13

“ПОГОДЖЕНО”

Керівник роботи:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.М. Головченко

Виконавець:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Коробка І.О.

Київ – 2021 року

(Обсяг програми (документа) , арк., Кб)

*32 арк., 1.6 мб*

*DVD-R*

(Вид носія даних)

***Тексти програмного коду***

***Програмне забезпечення для Веб-гра «Сапер» з можливістю авторизації та перегляду результатів ігор***

(Найменування програми (документа))

Київ – 2021

**СЕРВЕРНА ЧАСТИНА**

const {Game, User} = require('../models/index')

class GameController {

async create(req, res) {

const {isWin, level, timeAmount, date, percentageOfPassing, userId} = req.body

const game = await Game.create({isWin, level, timeAmount, date, percentageOfPassing, userId})

return res.json(game)

}

async delete(req, res) {

const {gameId} = req.body

const game = await Game.destroy({where: {id: gameId}})

return res.json(game)

}

async getWinGames(req, res) {

const games = await Game.findAll({

where: {isWin: true},

include: [{

model: User,

required: true

}]

})

return res.json(games)

}

async getAllGames(req, res) {

const games = await Game.findAll({

include: [{

model: User,

required: true

}]

})

return res.json(games)

}

}

module.exports = new GameController()

const {Game, User} = require('../models/index')

class GameController {

async create(req, res) {

const {isWin, level, timeAmount, date, percentageOfPassing, userId} = req.body

const game = await Game.create({isWin, level, timeAmount, date, percentageOfPassing, userId})

return res.json(game)

}

async delete(req, res) {

const {gameId} = req.body

const game = await Game.destroy({where: {id: gameId}})

return res.json(game)

}

async getWinGames(req, res) {

const games = await Game.findAll({

where: {isWin: true},

include: [{

model: User,

required: true

}]

})

return res.json(games)

}

async getAllGames(req, res) {

const games = await Game.findAll({

include: [{

model: User,

required: true

}]

})

return res.json(games)

}

}

module.exports = new GameController()

const jwt = require('jsonwebtoken')

module.exports = function (req, res, next) {

if (req.method === "OPTIONS") {

next()

}

try {

const token = req.headers.authorization.split(' ')[1]

if (!token) {

return res.status(401).json({message: "Не авторизован"})

}

req.user = jwt.verify(token, process.env.SECRET\_KEY)

next()

} catch (e) {

res.status(401).json({message: "Не авторизован"})

}

};

const jwt = require('jsonwebtoken')

module.exports = function(role) {

return function (req, res, next) {

if (req.method === "OPTIONS") {

next()

}

try {

const token = req.headers.authorization.split(' ')[1]

if (!token) {

return res.status(401).json({message: "Не авторизован"})

}

const decoded = jwt.verify(token, process.env.SECRET\_KEY)

if (decoded.role !== role) {

return res.status(403).json({message: "Нет доступа"})

}

req.user = decoded;

next()

} catch (e) {

res.status(401).json({message: "Не авторизован"})

}

};

}

const sequelize = require('../db')

const {DataTypes} = require('sequelize')

const User = sequelize.define('user', {

id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},

username: {type: DataTypes.STRING, unique: true,},

password: {type: DataTypes.STRING},

role: {type: DataTypes.STRING, defaultValue: "USER"},

raiting: {type: DataTypes.INTEGER, defaultValue: 0},

})

const Game = sequelize.define('game', {

id: {type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true},

isWin: {type: DataTypes.BOOLEAN, allowNull: false},

level: {type: DataTypes.STRING, allowNull: false},

timeAmount: {type: DataTypes.INTEGER, allowNull: false},

date: {type: DataTypes.DATE, allowNull: false},

percentageOfPassing: {type: DataTypes.INTEGER, allowNull: false},

})

User.hasMany(Game)

Game.belongsTo(User)

module.exports = {

User,

Game

}

const Router = require("express");

const router = new Router();

const gameController = require("../controllers/gameController");

const checkRole = require("../middleware/checkRole");

const auth = require("../middleware/auth");

router.post("/", auth, gameController.create);

router.get("/", auth, gameController.getWinGames);

router.get("/all", auth, gameController.getAllGames);

router.delete("/", checkRole("ADMIN"), gameController.delete);

module.exports = router;

const Router = require('express')

const router = new Router()

const userRouter = require('./userRouter')

const gameRouter = require('./gameRouter')

router.use('/user', userRouter)

router.use('/game', gameRouter)

module.exports = router

const Router = require('express')

const router = new Router()

const userController = require('../controllers/userController')

const auth = require('../middleware/auth')

router.post('/registration', userController.registration)

router.post('/login', userController.login)

router.get('/auth', auth, userController.check)

module.exports = router

require('dotenv').config()

const express = require('express')

const models = require('./models')

const cors = require('cors')

const router = require('./routes/')

const sequelize = require('./db')

const PORT = process.env.PORT || 5000

const app = express()

app.use(cors())

app.use(express.json())

app.use('/api', router)

const start = async () => {

try {

await sequelize.authenticate()

await sequelize.sync()

app.listen(PORT, () => console.log(`Server started on ${PORT}`))

} catch (error) {

console.log(error)

}

}

start()

require('dotenv').config()

const express = require('express')

const models = require('./models')

const cors = require('cors')

const router = require('./routes/')

const sequelize = require('./db')

const PORT = process.env.PORT || 5000

const app = express()

app.use(cors())

app.use(express.json())

app.use('/api', router)

const start = async () => {

try {

await sequelize.authenticate()

await sequelize.sync()

app.listen(PORT, () => console.log(`Server started on ${PORT}`))

} catch (error) {

console.log(error)

}

}

start()

**КЛІЄНТСЬКА ЧАСТИНА**

import React from "react";

import ReactDOM from "react-dom";

import App from "./App";

import {BrowserRouter} from "react-router-dom";

ReactDOM.render(

<React.StrictMode>

<BrowserRouter>

<App/>

</BrowserRouter>

</React.StrictMode>,

document.getElementById("root")

);

import "./App.css";

import { useEffect, useState } from "react";

import styled from "@emotion/styled";

import { Link, Route, Router, Switch, useHistory } from "react-router-dom";

import {

Divider,

List,

ListItem,

ListItemButton,

ListItemIcon,

ListItemText,

} from "@mui/material";

import InboxIcon from "@mui/icons-material/Inbox";

import DraftsIcon from "@mui/icons-material/Drafts";

import TopList from "./components/TopList";

import Auth from "./components/Auth";

import Admin from "./components/Admin";

import Home from "./components/Home";

import ContextAuth from "./context/contextAuth";

import { check } from "./http/authApi";

import Game from "./components/game";

const AppStyles = styled.div`

display: flex;

background: #ededed;

> section {

width: 100%;

height: 100vh;

background: #3d3d3d;

overflow: scroll;

display: flex;

justify-content: center;

align-items: center;

}

`;

const SideBarStyles = styled.aside`

width: 20%;

min-width: 300px;

overflow-y: scroll;

`;

const ADMIN\_ROUTE = "/admin";

const AUTH\_ROUTE = "/auth";

const HOME\_ROUTE = "/";

const GAME\_ROUTE = "/game";

const TOP\_LIST\_ROUTE = "/top-list";

const nav = [

{

showOnlyAuth: false,

showOnlyAdmin: false,

label: "HOME",

route: "/",

},

{

showOnlyAuth: true,

showOnlyAdmin: false,

label: "GAME",

route: "/game",

},

{

showOnlyAuth: true,

showOnlyAdmin: false,

label: "TOP LIST",

route: "/top-list",

},

{

showOnlyAuth: false,

showOnlyAdmin: false,

label: "ACCOUNT",

route: "/auth",

},

{

showOnlyAuth: true,

showOnlyAdmin: true,

label: "ADMIN",

route: "/admin",

},

];

const ItemNav = ({ item }) => (

<Link to={item.route}>

<ListItem disablePadding>

<ListItemButton>

<ListItemIcon>

<InboxIcon />

</ListItemIcon>

<ListItemText primary={item.label} />

</ListItemButton>

</ListItem>

</Link>

);

function App() {

const history = useHistory();

const [isAuth, setIsAuth] = useState(false);

const [user, setUser] = useState();

useEffect(() => {

check().then((resp) => {

setUser(resp);

setIsAuth(true);

});

}, []);

return (

<ContextAuth.Provider value={{ isAuth, user, setIsAuth, setUser }}>

<Router history={history}>

<AppStyles className="App">

<SideBarStyles>

<List>

{nav.map((item) => {

if (isAuth && item.showOnlyAuth) {

if (user && user.role === "ADMIN") {

return <ItemNav item={item} />;

}

if (!item.showOnlyAdmin) {

return <ItemNav item={item} />;

}

} else if (!item.showOnlyAuth) {

return <ItemNav item={item} />;

}

})}

</List>

<Divider />

<Switch>

{isAuth && [

<Route

key={ADMIN\_ROUTE}

path={ADMIN\_ROUTE}

component={Admin}

exact

/>,

<Route

key={TOP\_LIST\_ROUTE}

path={TOP\_LIST\_ROUTE}

component={TopList}

exact

/>,

]}

<Route key={AUTH\_ROUTE} path={AUTH\_ROUTE} component={Auth} exact />

</Switch>

</SideBarStyles>

<section>

<Switch>

{isAuth && (

<Route key={GAME\_ROUTE} path={GAME\_ROUTE} component={Game} exact />

)}

<Route key={HOME\_ROUTE} path={HOME\_ROUTE} component={Home} exact />

</Switch>

</section>

</AppStyles>

</Router>

</ContextAuth.Provider>

);

}

export default App;

export const OpenAllEmpty = (x, y, grid, settings) => {

let newGrid = JSON.parse(JSON.stringify(grid))

if (newGrid[x][y].isOpen) return newGrid

newGrid[x][y].isOpen = true;

const minX = x === 0 ? 0 : -1

const minY = y === 0 ? 0 : -1

const maxX = x === settings.rows - 1 ? 0 : 1

const maxY = y === settings.columns - 1 ? 0 : 1

for (let i = minX; i <= maxX; i++) {

for (let j = minY; j <= maxY; j++) {

if (!newGrid[x + i][y + j].isOpen && newGrid[x + i][y + j].minesAround === 0) {

newGrid = OpenAllEmpty(x + i, y + j, newGrid, settings);

} else if (newGrid[x + i][y + j].minesAround > 0) {

newGrid[x + i][y + j].isOpen = true;

}

}

}

return newGrid

}

export const Prompt = (grid, settings) => {

let newGrid = JSON.parse(JSON.stringify(grid))

for (let x = 0; x < settings.rows; x++) {

for (let y = 0; y < settings.columns; y++) {

if (grid[x][y].isOpen) {

const minX = x === 0 ? 0 : -1

const minY = y === 0 ? 0 : -1

const maxX = x === settings.rows - 1 ? 0 : 1

const maxY = y === settings.columns - 1 ? 0 : 1

for (let i = minX; i <= maxX; i++) {

for (let j = minY; j <= maxY; j++) {

if (!grid[x + i][y + j].isOpen && !newGrid[x + i][y + j].isMine) {

if (newGrid[x + i][y + j].minesAround === 0) {

return OpenAllEmpty(x + i, y + j, grid, settings)

} else {

newGrid[x + i][y + j].isOpen = true;

return newGrid

}

}

}

}

}

}

}

}

export const OpenAllMines = (grid, settings) => {

for (let x = 0; x < settings.rows; x++) {

for (let y = 0; y < settings.columns; y++) {

if (grid[x][y].isMine) {

grid[x][y].isOpen = true

}

}

}

return grid

}

export const random = (number) => Math.floor(Math.random() \* number);

export const createEmptyField = (columns, rows) => {

const field = [];

for (let x = 0; x < rows; x++) {

const col = [];

for (let y = 0; y < columns; y++) {

col.push({

x,

y,

minesAround: 0,

isFlag: false,

isOpen: false,

isMine: false,

});

}

field.push(col);

}

return field;

};

const createMines = (field, settings, currentClick) => {

const {columns, rows, minesCount} = settings;

let count = 0;

while (count < minesCount) {

const x = random(rows);

const y = random(columns);

if (!field[x][y].isMine) {

if ((y !== currentClick.y || x !== currentClick.x) &&

(y !== currentClick.y - 1 || x !== currentClick.x) &&

(y !== currentClick.y + 1 || x !== currentClick.x) &&

(y !== currentClick.y || x !== currentClick.x - 1) &&

(y !== currentClick.y || x !== currentClick.x + 1) &&

(y !== currentClick.y - 1 || x !== currentClick.x - 1) &&

(y !== currentClick.y + 1 || x !== currentClick.x + 1) &&

(y !== currentClick.y + 1 || x !== currentClick.x - 1) &&

(y !== currentClick.y - 1 || x !== currentClick.x + 1)) {

count++;

field[x][y].isMine = true;

}

}

}

return field;

};

const calcMinesRoundAllCells = (field, settings) => {

const {columns, rows} = settings;

for (let x = 0; x < rows; x++) {

for (let y = 0; y < columns; y++) {

let count = 0;

if (!field[x][y].isMine) {

x - 1 >= 0 && y - 1 >= 0 && field[x - 1][y - 1].isMine && count++;

y - 1 >= 0 && field[x][y - 1].isMine && count++;

x - 1 >= 0 && field[x - 1][y].isMine && count++;

y + 1 < columns && field[x][y + 1].isMine && count++;

x + 1 < rows && field[x + 1][y].isMine && count++;

x + 1 < rows && y + 1 < columns && field[x + 1][y + 1].isMine && count++;

x - 1 >= 0 && y + 1 < columns && field[x - 1][y + 1].isMine && count++;

x + 1 < rows && y - 1 >= 0 && field[x + 1][y - 1].isMine && count++;

if (count !== 0) {

field[x][y].minesAround = count;

}

}

}

}

return field;

};

export const createField = (grid, settings, currentClick) => {

const minesField = createMines(grid, settings, currentClick);

return calcMinesRoundAllCells(minesField, settings);

};

export const checkWin = (grid, settings) => {

for (let x = 0; x < settings.rows; x++) {

for (let y = 0; y < settings.columns; y++) {

if (!grid[x][y].isOpen && !grid[x][y].isMine ) {

return false

}

}

}

return true

}

import {$authHost} from "./index";

export const apiCreateGame = async (game) => {

const {data} = await $authHost.post('api/game', game )

return data

}

export const apiGetGames = async () => {

const {data} = await $authHost.get('api/game' )

return data

}

export const apiDeleteGame = async (gameId) => {

const {data} = await $authHost.delete('api/game/', {data: {gameId}})

return data

}

export const apiGetAllGames = async () => {

const {data} = await $authHost.get('api/game/all' )

return data

}

import axios from "axios";

const $host = axios.create({

baseURL: "http://localhost:8000/"

})

const $authHost = axios.create({

baseURL: "http://localhost:8000/"

})

const authInterceptor = config => {

config.headers.authorization = `Bearer ${localStorage.getItem('token')}`

return config

}

$authHost.interceptors.request.use(authInterceptor)

export {

$host,

$authHost

}

import {$authHost, $host} from "./index";

import jwt\_decode from "jwt-decode";

export const registration = async (username, password) => {

const {data} = await $host.post('api/user/registration', {username, password})

localStorage.setItem('token', data.token)

return jwt\_decode(data.token)

}

export const login = async (username, password) => {

const {data} = await $host.post('api/user/login', {username, password})

localStorage.setItem('token', data.token)

return jwt\_decode(data.token)

}

export const check = async () => {

const {data} = await $authHost.get('api/user/auth' )

localStorage.setItem('token', data.token)

return jwt\_decode(data.token)

}

import {createContext} from "react";

const ContextAuth = createContext()

export default ContextAuth

import styled from "@emotion/styled";

import StarIcon from '@mui/icons-material/Star';

import {List, ListItem, ListItemIcon, ListItemText} from "@mui/material";

import {useEffect, useState} from "react";

import {apiGetGames} from "../http/ss2";

const TopListStyles = styled.section`

padding: 0 20px;

li {

> span {

width: 100%;

> h3 {

margin: 0;

font-family: sans-serif;

}

> div {

margin-top: 5px;

display: flex;

justify-content: space-between;

> h4 {

margin: 0;

font-weight: 400;

font-size: 0.8rem;

font-family: sans-serif;

}

> span {

font-size: 0.8rem;

font-family: sans-serif;

}

}

}

}

`

const TopList = () => {

const [list, setList] = useState([])

useEffect(() => {

const fetchData = async () => {

const resp = await apiGetGames()

setList(resp)

}

fetchData();

}, [])

return (

<TopListStyles>

<h2>Top results</h2>

<List>

{list.map((item, index) => (

<ListItem alignItems="flex-start">

{index < 3 && (

<ListItemIcon>

<StarIcon/>

</ListItemIcon>

)}

<span>

<h3>{item.user.username}</h3>

<div>

<h4>{item.level}</h4>

<span>{("0" + Math.floor((item.timeAmount / 1000) % 60)).slice(-2)}</span>

</div>

</span>

</ListItem>

))}

</List>

</TopListStyles>

)

}

export default TopList;

import styled from "@emotion/styled";

const HomeStyles = styled.section`

width: 100%;

box-sizing: border-box;

padding: 100px;

color: #fff;

> h1 {

margin-top: 50px;

font-size: 2.5rem;

font-family: sans-serif;

font-weight: 700;

text-transform: uppercase;

}

> h3 {

font-size: 1.7rem;

font-family: sans-serif;

font-weight: 700;

text-transform: uppercase;

}

> p {

font-size: 1.2rem;

font-family: sans-serif;

font-weight: 400;

}

`;

const Home = () => {

return (

<HomeStyles>

<h1>Welcome to the game minesweeper</h1>

<p>

Minesweeper is a game that brings nostalgia to the 80s and early 90s

children. It was first available on the family computer, back when computers

were huge and not as mobile as your laptop or smartphones today.

</p>

<p>

So, there is a desk in most homes wherein people share a clunky device. And

when some children during that time got their chance to sit in front of its

screen -- one of the games played is the Minesweeper. It was before the

internet was a thing you consume the whole day. Offline was still the norm.

</p>

<h3>How To Play It the Right Way?</h3>

<p>

To play Minesweeper, players have to press a square first. But it is more

than just guessing which square to click. Observation and deduction is the

name of the game.

</p>

<p>

Take note that the number around the boxes is the clue to know where the

mines are.

</p>

<p>

There are ones and twos, and it will tell you how many bombs are touching the

square. If the box is near number one -- it means that one of the squares

around it has the "mine." Likewise, the number two indicates that there is

more than one explosive adjacent to it.{" "}

</p>

<p>

It turns out that these numbers guide players throughout the game. The final

goal is to clear the map and refrain from blowing the bomb.

</p>

</HomeStyles>

);

};

export default Home;

import {

Button,

ListItem,

ListItemButton,

ListItemIcon,

ListItemText,

TextField,

} from "@mui/material";

import { useContext, useRef, useState } from "react";

import styled from "@emotion/styled";

import { Link, useHistory, useLocation } from "react-router-dom";

import ContextAuth from "../context/contextAuth";

import DraftsIcon from "@mui/icons-material/Drafts";

import { login, registration } from "../http/authApi";

const AuthStyles = styled.div`

display: flex;

flex-direction: column;

padding: 20px;

> h3 {

font-family: sans-serif;

font-weight: 500;

> span {

font-weight: bold;

}

}

> div {

margin-top: 20px;

}

> footer {

margin-top: 20px;

display: flex;

justify-content: space-between;

> button {

width: calc(50% - 10px);

}

> span {

font-family: sans-serif;

font-size: 0.8rem;

text-decoration: underline;

font-weght: 700;

cursor: pointer;

}

}

`;

const Auth = () => {

const { isAuth, user, setUser, setIsAuth } = useContext(ContextAuth);

const [authType, setAuthType] = useState("LOGIN");

const [username, setUsername] = useState();

const [password, setPassword] = useState();

const authHandle = async () => {

try {

let resp;

if (authType === "LOGIN") {

resp = await login(username, password);

} else {

resp = await registration(username, password);

}

setUser(resp);

setIsAuth(true);

} catch (e) {

alert(e);

}

};

return (

<AuthStyles>

{isAuth ? (

<>

<h3>

<span>User:</span> {user?.username}

</h3>

<Button

variant="contained"

onClick={() => {

setUser({});

localStorage.removeItem("token");

setIsAuth(false);

}}

>

Log Out

</Button>

</>

) : (

<>

<TextField

id="demo-helper-text-aligned"

value={username}

onChange={(e) => setUsername(e.target.value)}

type="username"

label="Username"

/>

<TextField

id="demo-helper-text-aligned"

value={password}

onChange={(e) => setPassword(e.target.value)}

type="password"

label="Password"

/>

{authType === "LOGIN" ? (

<footer>

<Button variant="contained" onClick={authHandle}>

Login

</Button>

<span

onClick={() => {

setAuthType("REGISTRATION");

}}

>

Sign Up

</span>

</footer>

) : (

<footer>

<Button variant="contained" onClick={authHandle}>

Sign Up

</Button>

<span

onClick={() => {

setAuthType("LOGIN");

}}

>

Login

</span>

</footer>

)}

</>

)}

</AuthStyles>

);

};

export default Auth;

import styled from "@emotion/styled";

import {IconButton, List, ListItem, ListItemIcon,} from "@mui/material";

import DeleteIcon from '@mui/icons-material/Delete';

import {useEffect, useState} from "react";

import {apiDeleteGame, apiGetAllGames, apiGetGames} from "../http/ss2";

const TopListStyles = styled.section`

padding: 0 20px;

li {

display: flex;

align-items: center;

border-bottom: 1px solid black;

>div {

margin-top: 0;

}

> span {

width: 100%;

> h3 {

margin: 0;

font-family: sans-serif;

}

> div {

margin-top: 5px;

display: flex;

justify-content: space-between;

> h4 {

margin: 0;

font-weight: 400;

font-size: 0.8rem;

font-family: sans-serif;

}

> span {

font-size: 0.8rem;

font-family: sans-serif;

}

}

}

}

`

const TopList = () => {

const [list, setList] = useState([])

useEffect(() => {

const fetchData = async () => {

const resp = await apiGetAllGames()

setList(resp)

}

fetchData();

}, [])

const deleteItem = async (gameId) => {

await apiDeleteGame(gameId)

setList(list.filter(item => item.id !== gameId))

}

return (

<TopListStyles>

<h2>Games</h2>

<List>

{list.map((item) => (

<ListItem alignItems="flex-start">

<ListItemIcon>

<IconButton aria-label="delete" onClick={() => deleteItem(item.id)}>

<DeleteIcon />

</IconButton>

</ListItemIcon>

<span>

<h3>{item.user.username}</h3>

<div>

<h4>{item.level}</h4>

<span>{("0" + Math.floor((item.timeAmount / 1000) % 60)).slice(-2)}</span>

</div>

</span>

</ListItem>

))}

</List>

</TopListStyles>

)

}

export default TopList;

import {useState, useEffect} from "react";

const StopWatch = ({isStart, isGameOver,}) => {

const [isActive, setIsActive] = useState(false);

const [time, setTime] = useState(0);

useEffect(() => {

let interval = null;

if (isActive) {

interval = setInterval(() => {

setTime((time) => time + 10);

}, 10);

} else {

clearInterval(interval);

}

return () => {

clearInterval(interval);

};

}, [isActive]);

useEffect(() => {

setIsActive(isStart);

setTime(0);

}, [isStart])

useEffect(() => {

setIsActive(false);

}, [isGameOver])

return (

<div>

<span>

{("0" + Math.floor((time / 60000) % 60)).slice(-2)}:

</span>

<span>

{("0" + Math.floor((time / 1000) % 60)).slice(-2)}

</span>

</div>

);

}

export default StopWatch;

import {Button, ToggleButton, ToggleButtonGroup} from "@mui/material";

import styled from "@emotion/styled";

import {useEffect, useState} from "react";

import StopWatch from "./StopWatch";

import Field from "./Field";

import AccessAlarmIcon from '@mui/icons-material/AccessAlarm';

import TourIcon from '@mui/icons-material/Tour';

const GameStyles = styled.section`

button {

color: #fff;

border-color: #fff;

}

> header {

color: #fff;

display: flex;

background: #006d77;

padding: 10px 20px;

border-radius: 10px 10px 0 0;

justify-content: space-between;

position: relative;

box-shadow: 0 0 5px #000;

z-index: 9;

> span {

display: flex;

align-items: center;

> span {

display: flex;

align-items: center;

font-family: sans-serif;

font-weight: bold;

margin-right: 20px;

&:last-child {

margin-right: 0;

}

> .watch {

margin-right: 10px;

height: 25px;

}

}

}

}

> section {

display: flex;

flex-direction: column;

}

> footer {

padding: 10px 20px 10px 70px;

box-shadow: 0 0 5px #000;

background: #006d77;

border-radius: 0 0 10px 10px;

display: flex;

justify-content: space-between;

align-items: center;

font-family: sans-serif;

font-weight: bold;

color: #fff;

}

`

const settingTypes = {

senior: {

name: "SENIOR",

columns: 30,

rows: 20,

minesCount: 100,

},

middle: {

name: "MIDDLE",

columns: 20,

rows: 15,

minesCount: 45,

},

junior: {

name: "JUNIOR",

columns: 20,

rows: 10,

minesCount: 15,

},

}

const Game = () => {

const [isStart, setIsStart] = useState(false);

const [isGameOver, setIsGameOver] = useState(false);

const [settings, setSettings] = useState(settingTypes.junior);

const [minesLeft, setMinesLeft] = useState();

const [isWin, setIsWin] = useState(false);

useEffect(() => {

setMinesLeft(settings.minesCount)

setIsStart(false)

setIsGameOver(false)

setIsWin(false)

}, [settings])

return (

<GameStyles>

<header>

<ToggleButtonGroup

size="small"

value={settings}

exclusive

onChange={(event, newAlignment) => setSettings(newAlignment.settings)}

>

{Object.values(settingTypes).map(settings => (

<ToggleButton size="small" value={{settings}}>

{settings.name}

</ToggleButton>

))}

</ToggleButtonGroup>

<span>

<span><span>{minesLeft}</span><TourIcon/></span>

<span><StopWatch isStart={isStart} isGameOver={isGameOver}/><AccessAlarmIcon/></span>

</span>

</header>

<Field setMinesLeft={setMinesLeft} minesLeft={minesLeft} setSettings={setSettings} settings={settings}

setIsWin={setIsWin} isWin={isWin} setIsGameOver={setIsGameOver} isGameOver={isGameOver}

setIsStart={setIsStart} isStart={isStart}/>

<footer>

<Button size="small" variant="outlined" onClick={() => setSettings({...settings})}>New Game</Button>

{isGameOver && <span>{isWin ? "You are win!" : "You are lose :("}</span>}

</footer>

</GameStyles>

)

}

export default Game

import { useContext, useEffect, useState } from "react";

import { createEmptyField, createField } from "../../util/createField";

import { CircularProgress, IconButton } from "@mui/material";

import styled from "@emotion/styled";

import Cell from "./Cell";

import { OpenAllEmpty, OpenAllMines, Prompt } from "../../util/reveal";

import { checkWin } from "../../util/checkWin";

import { apiCreateGame } from "../../http/ss2";

import ContextAuth from "../../context/contextAuth";

import EmojiObjectsIcon from "@mui/icons-material/EmojiObjects";

const FieldStyles = styled.section`

display: flex;

flex-direction: column;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.7);

position: relative;

> ul {

display: flex;

margin: 0;

padding: 0;

list-style: none;

}

> button {

position: absolute;

bottom: -5px;

left: 20px;

transform: translateY(100%);

}

`;

const Field = ({

setIsWin,

isGameOver,

setIsGameOver,

setIsStart,

isStart,

settings,

minesLeft,

setMinesLeft,

}) => {

const { user } = useContext(ContextAuth);

const [grid, setGrid] = useState([]);

const [timeStart, setTimeStart] = useState();

useEffect(() => {

const newGrid = createEmptyField(settings.columns, settings.rows);

setGrid(newGrid);

}, [settings]);

useEffect(() => {

if (isStart) {

setTimeStart(new Date());

}

}, [isStart]);

const prompt = () => {

if (isStart && !isGameOver) {

const newGrid = Prompt(grid, settings);

if (newGrid) {

setGrid(newGrid);

}

}

};

const updFlag = (e, x, y) => {

e.preventDefault();

if (!isStart || isGameOver || grid[x][y].isOpen) return;

const newGrid = JSON.parse(JSON.stringify(grid));

newGrid[x][y].isFlag = !newGrid[x][y].isFlag;

setMinesLeft((minesLeft) => minesLeft + (1 \* newGrid[x][y].isFlag ? -1 : 1));

checkGameOver(newGrid);

setGrid(newGrid);

};

const checkGameOver = async (newGrid, gameOver = false) => {

let checkIsWin = checkWin(newGrid, settings);

setIsWin(checkIsWin);

if (gameOver || checkIsWin) {

setIsGameOver(true);

setMinesLeft(settings.minesCount);

const dataNow = new Date();

await apiCreateGame({

isWin: checkIsWin,

level: settings.name,

timeAmount: dataNow - timeStart,

date: dataNow,

percentageOfPassing: Math.round(

(100 \* (settings.minesCount - minesLeft)) / settings.minesCount

),

userId: user.id,

});

} else {

setIsGameOver(false);

}

};

const updOpen = (x, y) => {

if (!isStart) {

const newGrid = createField(grid, settings, { x, y });

setGrid(newGrid);

setIsStart(true);

}

if (grid[x][y].isOpen || grid[x][y].isFlag || isGameOver) return;

let newGrid = JSON.parse(JSON.stringify(grid));

if (newGrid[x][y].minesAround > 0) {

newGrid[x][y].isOpen = true;

} else if (newGrid[x][y].isMine) {

newGrid = OpenAllMines(newGrid, settings);

checkGameOver(newGrid, true);

setGrid(newGrid);

return;

} else if (newGrid[x][y].minesAround === 0) {

newGrid = OpenAllEmpty(x, y, newGrid, settings);

}

checkGameOver(newGrid);

setGrid(newGrid);

};

if (!grid) return <CircularProgress />;

return (

<FieldStyles>

{grid.map((row) => (

<ul key={`${JSON.stringify(row)}`}>

{row.map((cell) => (

<Cell

key={`${JSON.stringify(cell)}-${JSON.stringify(row)}`}

cell={cell}

updFlag={updFlag}

updOpen={updOpen}

isGameOver={isGameOver}

/>

))}

</ul>

))}

<IconButton color="secondary" onClick={prompt}>

<EmojiObjectsIcon />

</IconButton>

</FieldStyles>

);

};

export default Field;

import styled from "@emotion/styled";

import {random} from "../../util/createField";

const mineColors = ["#ff0040", "#ff9500", "#ffff00", "#00ff15", "#00ffff", "#0095ff", "#6a00ff", "#ff00ff"]

const numberColors = ["#000", "#00a6fb", "#7FB800", "#53610D", "#805D93", "#ff6d00", "#032B43", "#780116"]

const CellStyles = styled.li`

display: flex;

position: relative;

width: 20px;

height: 20px;

justify-content: center;

background: ${({cell}) => !cell.isOpen ? "#3d3d3d" : "#eeeeee"};

align-items: center;

background: ${({cell}) => {

if (cell.isOpen && cell.isMine) return mineColors[random(mineColors.length)]

if (cell.isOpen && cell.x % 2 === cell.y % 2) return "#f9dcc4";

if (cell.isOpen && cell.x % 1 === cell.y % 1) return "#fec89a";

if (!cell.isOpen && cell.x % 2 === cell.y % 2) return "#006d77";

if (!cell.isOpen && cell.x % 1 === cell.y % 1) return "#83c5be";

}};

box-shadow: 0 0 1px black;

color: ${({cell}) => numberColors[cell.minesAround]};

font-family: sans-serif;

font-weight: bold;

&:after {

content: '';

display: ${({cell}) => cell.isMine && cell.isOpen ? "flex" : "none"};

position: absolute;

top: 50%;

left: 50%;

border-radius: 50%;

transform: translate(-50%, -50%);

width: 50%;

height: 50%;

background: rgba(0, 0, 0, .3);

}

> img {

width: 100%;

height: 100%;

}

`;

// background: ${({cell}) => !cell.isOpen ? cell.x % 2 === cell.y % 2 ? "#2196f3" : "#42a5f5" : "#eeeeee"};

// background: ${({cell}) => !cell.isOpen ? cell.x % 2 === cell.y % 2 ? "#2a850e" : "#27a300" : "#eeeeee"};

// background: ${({cell}) => !cell.isOpen ? cell.x % 2 === cell.y % 2 ? "#4f0099" : "#5c00b3" : "#eeeeee"};

// background: ${({cell}) => !cell.isOpen ? cell.x % 2 === cell.y % 2 ? "#7c98b3" : "#accbe1" : "#eeeeee"};

const Cell = ({cell, updFlag, updOpen, isGameOver}) => {

return (

<CellStyles

cell={cell}

onClick={() => updOpen(cell.x, cell.y)}

onContextMenu={(e) => updFlag(e, cell.x, cell.y)}

>

{cell.isOpen && cell.minesAround > 0 && cell.minesAround}

{!isGameOver && cell.isFlag && <img src="./flag.png" alt="f" />}

</CellStyles>

)

}

export default Cell;