**Приложение**

**Приложение 1**

**Код в файле «server.js»**

const express = require("express");

const cors = require("cors");

const app = express();

var corsOptions = {

    origin: "http://localhost:8081"

};

app.use(cors(corsOptions));

app.use(express.json());

app.use(express.urlencoded({ extended: true }));

const db = require("../nodejs-express-sequelize-mysql/models");

db.sequelize.sync()

    .then(() => {

        console.log("Synced db.");

    })

    .catch((err) => {

        console.log("Failed to sync db: " + err.message);

    });

app.get("/", (req, res) => {

    res.json({ message: "Welcome to my application." });

});

//routes

require("../nodejs-express-sequelize-mysql/routes/master.routes")(app);

require("../nodejs-express-sequelize-mysql/routes/service.routes")(app);

require("../nodejs-express-sequelize-mysql/routes/group.routes")(app);

require("../nodejs-express-sequelize-mysql/routes/shedule.routes")(app);

require('../nodejs-express-sequelize-mysql/routes/auth.routes')(app);

require('../nodejs-express-sequelize-mysql/routes/user.routes')(app);

const PORT = process.env.PORT || 8080;

app.listen(PORT, () => {

    console.log(`Server is running on port ${PORT}.`);

});

**Приложение 2**

**Код в файле «db.config.js»**

*module.exports* = {

  HOST: "localhost",

  USER: "root",

  PASSWORD: "Alias12345",

  DB: "diplomdb",

  dialect: "mysql",

  pool: {

    max: 5,

    min: 0,

    acquire: 30000,

    idle: 10000

  }

};

**Приложение 3**

**Код в файле «index.js»**

// инициализация sequelize

const dbConfig = require("../config/db.config.js");

const Sequelize = require("sequelize");

const sequelize = new Sequelize(dbConfig.DB, dbConfig.USER, dbConfig.PASSWORD, {

  host: dbConfig.HOST,

  dialect: dbConfig.dialect,

  operatorsAliases: 0,

  pool: {

    max: dbConfig.pool.max,

    min: dbConfig.pool.min,

    acquire: dbConfig.pool.acquire,

    idle: dbConfig.pool.idle

  }

});

const db = {};

db.Sequelize = Sequelize;

db.sequelize = sequelize;

db.masters = require("./master.model.js")(sequelize, Sequelize);

db.services = require("./service.model.js")(sequelize, Sequelize);

db.group\_services = require("./group.model.js")(sequelize, Sequelize);

db.shedules = require("./shedule.model.js")(sequelize, Sequelize);

db.user = require("../models/user.model.js")(sequelize, Sequelize);

db.role = require("../models/role.model.js")(sequelize, Sequelize);

db.role.belongsToMany(db.user, {

  through: "user\_roles"

});

db.user.belongsToMany(db.role, {

  through: "user\_roles"

});

db.ROLES = ["user", "admin", "moderator"];

db.user.hasMany(db.shedules);

db.shedules.belongsTo(db.user);

db.masters.hasMany(db.shedules);

db.shedules.belongsTo(db.masters);

db.services.hasMany(db.shedules);

db.shedules.belongsTo(db.services);

module.exports = db;

**Приложение 4**

**Описание моделей Sequelize**

**Код в файле «master.model.js»**

*module.exports* = (*sequelize*, *Sequelize*) => {

const Master = *sequelize*.define("masters", {

master\_name: {

type: *Sequelize*.STRING

},

birthdate: {

type: *Sequelize*.STRING

},

phone: {

type: *Sequelize*.STRING

},

position: {

type: *Sequelize*.STRING

},

photo: {

type: *Sequelize*.STRING

},

about\_text: {

type: *Sequelize*.STRING

},

group\_service\_id: {

type: *Sequelize*.INTEGER

}

})

return Master;

};

**Код в файле «service.model.js»**

*module.exports* = (*sequelize*, *Sequelize*) => {

const Service = *sequelize*.define("services", {

title: {

type: *Sequelize*.STRING

},

group\_service\_id: {

type: *Sequelize*.INTEGER

},

duration: {

type: *Sequelize*.INTEGER

},

price: {

type: *Sequelize*.INTEGER

}

});

return Service;

};

**Код в файле «role.model.js»**

module.exports = (sequelize, Sequelize) => {

    const Role = sequelize.define("roles", {

      id: {

        type: Sequelize.INTEGER,

        primaryKey: true

      },

      name: {

        type: Sequelize.STRING

      }

    });

    return Role;

  };

**Код в файле «user.model.js»**

module.exports = (sequelize, Sequelize) => {

  const User = sequelize.define("users", {

    username: {

      type: Sequelize.STRING

    },

    email: {

      type: Sequelize.STRING

    },

    password: {

      type: Sequelize.STRING

    },

    name: {

      type: Sequelize.STRING

    },

    phone: {

      type: Sequelize.STRING

    },

    birthdate: {

      type: Sequelize.STRING

    },

  });

  return User;

};

**Код в файле «shedule.model.js»**

module.exports = (sequelize, Sequelize) => {

  const Shedule = sequelize.define("shedules", {

    date: {

      type: Sequelize.STRING

    },

    time: {

      type: Sequelize.STRING

    },

    userId: {

      type: Sequelize.INTEGER

    },

    masterId: {

      type: Sequelize.INTEGER

    },

    serviceId: {

      type: Sequelize.INTEGER

    }

  });

  return Shedule;

};

**Приложение 5**

**Содержимое папки «controllers»**

**Код в файле «master.controller.js»**

const db = require("../models");

const Master = db.masters;

const Op = db.Sequelize.Op;

exports.create = (req, res) => {

    if (!req.body.master\_name) {

        res.status(400).send({

            message: "Content can not be empty!"

        });

        return;

    }

    const master = {

        master\_name: req.body.master\_name,

        birthdate: req.body.birthdate,

        phone: req.body.phone,

        position: req.body.position,

        photo: req.body.photo,

        about\_text: req.body.about\_text,

        group\_service\_id: req.body.group\_service\_id

    };

    Master.create(master)

        .then(data => {

            res.send(data);

        })

        .catch(err => {

            res.status(500).send({

                message:

                    err.message || "Some error occurred while creating the Master."

            });

        });

};

exports.findAll = (req, res) => {

    const group\_service\_id = req.query.group\_service\_id;

    var condition = group\_service\_id ? { group\_service\_id: { [Op.eq]: group\_service\_id } } : null;

    Master.findAll({ where: condition })

        .then(data => {

            res.send(data);

        })

        .catch(err => {

            res.status(500).send({

                message:

                    err.message || "Some error occurred while retrieving masters."

            });

        });

};

exports.findOne = (req, res) => {

    const id = req.params.id;\

  Master.findByPk(id)

        .then(data => {

            if (data) {

                res.send(data);

            } else {

                res.status(404).send({

                    message: `Cannot find Master with id=${id}.`

                });

            }

        })

        .catch(err => {

            res.status(500).send({

                message: "Error retrieving Master with id=" + id

            });

        });

};

exports.findByNameAndPhone = (req, res) => {

    const name = req.query.name;

    const phone = req.query.phone;

    var condition = { [Op.and]: [{ master\_name: `${name}` }, { phone: phone }] };

    Master.findOne({ where: condition })

        .then(data => {

            res.send(data);

        })

        .catch(err => {

            res.status(500).send({

                message:

                    err.message || "Some error occurred while retrieving masters."

            });

        });

};

exports.delete = (req, res) => {

    const id = req.params.id;

    Master.destroy({

        where: { id: id }

    })

        .then(num => {

            if (num == 1) {

                res.send({

                    message: "Master was deleted successfully!"

                });

            } else {

                res.send({

                    message: `Cannot delete Master with id=${id}. Maybe Master was not found!`

                });

            }

        })

        .catch(err => {

            res.status(500).send({

                message: "Could not delete Master with id=" + id

            });

        });

};

exports.update = (req, res) => {

    const id = req.params.id;

    Master.update(req.body, {

        where: { id: id }

    })

        .then(num => {

            if (num == 1) {

                res.send({

                    message: "Master was updated successfully."

                });

            } else {

                res.send({

                    message: `Cannot update Master with id=${id}. Maybe MAster was not found or req.body is empty!`

                });

            }

        })

        .catch(err => {

            res.status(500).send({

                message: "Error updating Master with id=" + id

            });

        });

};

**Код в файле «service.controller.js»**

const db = require("../models");

const Service = db.services;

const Op = db.Sequelize.Op;

exports.create = (req, res) => {

    // Validate request

    if (!req.body.title) {

        res.status(400).send({

            message: "Content can not be empty!"

        });

        return;

    }

    const service = {

        title: req.body.title,

        group\_service\_id: req.body.group\_service\_id,

        duration: req.body.duration,

        price: req.body.price

    };

    Service.create(service)

        .then(data => {

            res.send(data);

        })

        .catch(err => {

            res.status(500).send({

                message:

                    err.message || "Some error occurred while creating the Service."

            });

        });

};

exports.findAll = (req, res) => {

    const title = req.query.title;

    var condition = title ? { title: { [Op.like]: `%${title}%` } } : null;

    Service.findAll({ where: condition })

        .then(data => {

            res.send(data);

        })

        .catch(err => {

            res.status(500).send({

                message:

                    err.message || "Some error occurred while retrieving Service."

            });

        });

};

exports.findServiceByGroup = (req, res) => {

    const group\_service\_id = req.query.group\_service\_id;

    Service.findAll({ where: { group\_service\_id: group\_service\_id, }, })

        .then(data => {

            res.send(data);

        })

        .catch(err => {

            res.status(500).send({

                message:

                    err.message || "Some error occurred while retrieving Service."

            });

        });

};

exports.findOne = (req, res) => {

    const id = req.params.id;

    Service.findByPk(id)

        .then(data => {

            if (data) {

                res.send(data);

            } else {

                res.status(404).send({

                    message: `Cannot find Service with id=${id}.`

                });

            }

        })

        .catch(err => {

            res.status(500).send({

                message: "Error retrieving Service with id=" + id

            });

        });

};

exports.update = (req, res) => {

    const id = req.params.id;

    Service.update(req.body, {

        where: { id: id }

    })

        .then(num => {

            if (num == 1) {

                res.send({

                    message: "Service was updated successfully."

                });

            } else {

                res.send({

                    message: `Cannot update Service with id=${id}. Maybe Service was not found or req.body is empty!`

                });

            }

        })

        .catch(err => {

            res.status(500).send({

                message: "Error updating Master with id=" + id

            });

        });

};

exports.delete = (req, res) => {

    const id = req.params.id;

    Service.destroy({

        where: { id: id }

    })

        .then(num => {

            if (num == 1) {

                res.send({

                    message: "Service was deleted successfully!"

                });

            } else {

                res.send({

                    message: `Cannot delete Service with id=${id}. Maybe Service was not found!`

                });

            }

        })

        .catch(err => {

            res.status(500).send({

                message: "Could not delete Service with id=" + id

            });

        });

};

**Код в файле «shedule.controller.js»**

const db = require("../models");

const Shedule = db.shedules;

const User = db.user;

const Master = db.masters;

const Service = db.services;

const Op = db.Sequelize.Op;

exports.create = (req, res) => {

    if (!req.body.date) {

        res.status(400).send({

            message: "Content can not be empty!"

        });

        return;

    }

    const shedule = {

        date: req.body.date,

        time: req.body.time,

        userId: req.body.userId,

        masterId: req.body.masterId,

        serviceId: req.body.serviceId

    };

    Shedule.create(shedule)

        .then(data => {

            res.send(data);

        })

        .catch(err => {

            res.status(500).send({

                message:

                    err.message || "Some error occurred while creating the shedule."

            });

        });

};

exports.findAll = (req, res) => {

    const date = req.query.date;

    const masterId = req.query.masterId;

    var condition = date && masterId ? { [Op.and]: [{ date: `${date}` }, { masterId: masterId }] } : null;

    Shedule.findAll({

        where: condition

    })

        .then(data => {

            res.send(data);

        })

        .catch(err => {

            res.status(500).send({

                message:

                    err.message || "Some error occurred while retrieving Shedule."

            });

        });

};

exports.findOne = (req, res) => {

    const id = req.params.id;

    Shedule.findByPk(id)

        .then(data => {

            if (data) {

                res.send(data);

            } else {

                res.status(404).send({

                    message: `Cannot find Shedule with id=${id}.`

                });

            }

        })

        .catch(err => {

            res.status(500).send({

                message: "Error retrieving Shedule with id=" + id

            });

        });

};

exports.findUserShedule = (req, res) => {

    const userId = req.query.userId;

    Shedule.findAll({

        where: { userId: userId }, include: [

            {

                model: Master,

                attributes: ['master\_name']

            },

            {

                model: Service,

                attributes: ['title', 'duration', 'price']

            }]

    })

        .then(data => {

            res.send(data);

        })

        .catch(err => {

            res.status(500).send({

                message:

                    err.message || "Some error occurred while retrieving Shedule."

            });

        });

};

exports.findAllSheduleForAdmin = (req, res) => {

    const date = req.query.date;

    Shedule.findAll({

        where: { date: date }, include: [{

            model: User,

            attributes: ['name', 'phone']

        },

        {

            model: Master,

            attributes: ['master\_name', 'position']

        },

        {

            model: Service,

            attributes: ['title', 'duration', 'price']

        }]

    })

        .then(data => {

            res.send(data);

        })

        .catch(err => {

            res.status(500).send({

                message:

                    err.message || "Some error occurred while retrieving Shedule."

            });

        });

};

exports.findAllSheduleForMaster = (req, res) => {

    const date = req.query.date;

    const masterId = req.query.masterId;

    var condition = date && masterId ? { [Op.and]: [{ date: `${date}` }, { masterId: masterId }] } : null;

    Shedule.findAll({

        where: condition, include: [{

            model: User,

            attributes: ['name']

        },

        {

            model: Service,

            attributes: ['title']

        }]

    })

        .then(data => {

            res.send(data);

        })

        .catch(err => {

            res.status(500).send({

                message:

                    err.message || "Some error occurred while retrieving Shedule."

            });

        });

};

exports.delete = (req, res) => {

    const id = req.params.id;

    Shedule.destroy({

        where: { id: id }

    })

        .then(num => {

            if (num == 1) {

                res.send({

                    message: "Shedule was deleted successfully!"

                });

            } else {

                res.send({

                    message: `Cannot delete Shedule with id=${id}. Maybe Shedule was not found!`

                });

            }

        })

        .catch(err => {

            res.status(500).send({

                message: "Could not delete User with id=" + id

            });

        });

};

**Код в файле «group.controller.js»**

const db = require("../models");

const Group = db.group\_services;

const Op = db.Sequelize.Op;

exports.create = (req, res) => {

    // Validate request

    if (!req.body.title) {

        res.status(400).send({

            message: "Content can not be empty!"

        });

        return;

    }

    const groupService = {

        title: req.body.title

    };

    Group.create(groupService)

        .then(data => {

            res.send(data);

        })

        .catch(err => {

            res.status(500).send({

                message:

                    err.message || "Some error occurred while creating the GroupService."

            });

        });

};

exports.findAll = (req, res) => {

    const title = req.query.title;

    var condition = title ? { title: { [Op.like]: `%${title}%` } } : null;

    Group.findAll({ where: condition })

        .then(data => {

            res.send(data);

        })

        .catch(err => {

            res.status(500).send({

                message:

                    err.message || "Some error occurred while retrieving GroupService."

            });

        });

};

exports.findOne = (req, res) => {

    const id = req.params.id;

    Group.findByPk(id)

        .then(data => {

            if (data) {

                res.send(data);

            } else {

                res.status(404).send({

                    message: `Cannot find GroupService with id=${id}.`

                });

            }

        })

        .catch(err => {

            res.status(500).send({

                message: "Error retrieving GroupService with id=" + id

            });

        });

};

**Код в файле «user.controller.js»**

const db = require("../models");

const User = db.user;

const Op = db.Sequelize.Op;

exports.allAccess = (req, res) => {

    res.status(200).send("Public Content.");

};

exports.userBoard = (req, res) => {

    res.status(200).send("User Content.");

};

exports.adminBoard = (req, res) => {

    res.status(200).send("Admin Content.");

};

exports.masterBoard = (req, res) => {

    res.status(200).send("Master Content.");

};

exports.findAll = (req, res) => {

    const start = +req.query.start;

    const count = +req.query.count;

    const phone = req.query.phone;

    var condition = phone ? { phone: { [Op.like]: `%${phone}%` } } : null;

    User.findAll({

        where: condition,

        offset: start, limit: count

    })

        .then(data => {

            res.send(data);

        })

        .catch(err => {

            res.status(500).send({

                message:

                    err.message || "Some error occurred while retrieving User."

            });

        });

};

exports.delete = (req, res) => {

    const id = req.params.id;

    User.destroy({

        where: { id: id }

    })

        .then(num => {

            if (num == 1) {

                res.send({

                    message: "User was deleted successfully!"

                });

            } else {

                res.send({

                    message: `Cannot delete User with id=${id}. Maybe User was not found!`

                });

            }

        })

        .catch(err => {

            res.status(500).send({

                message: "Could not delete User with id=" + id

            });

        });

};

exports.usersCount = (req, res) => {

    const phone = req.query.phone;

    var condition = phone ? { phone: { [Op.like]: `%${phone}%` } } : null;

    User.count({

        where: condition

    }).then(data => {

        res.send(String(data));

    })

        .catch(err => {

            res.status(500).send({

                message:

                    err.message || "Some error"

            });

        });

};

**Код в файле «auth.controller.js»**

const db = require("../models");

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

const User = db.user;

const Role = db.role;

const Op = db.Sequelize.Op;

var jwt = require("jsonwebtoken");

var bcrypt = require("bcryptjs");

exports.signup = (req, res) => {

    User.create({

        username: req.body.username,

        email: req.body.email,

        password: bcrypt.hashSync(req.body.password, 8),

        name: req.body.name,

        phone: req.body.phone,

        birthdate: req.body.birthdate

    })

        .then(user => {

            if (req.body.roles) {

                Role.findAll({

                    where: {

                        name: {

                            [Op.or]: req.body.roles

                        }

                    }

                }).then(roles => {

                    user.setRoles(roles).then(() => {

                        res.send({ message: "User was registered successfully!" });

                    });

                });

            } else {

                // user role = 1

                user.setRoles([1]).then(() => {

                    res.send({ message: "User was registered successfully!" });

                });

            }

        })

        .catch(err => {

            res.status(500).send({ message: err.message });

        });

};

exports.signin = (req, res) => {

    User.findOne({

        where: {

            username: req.body.username

        }

    })

        .then(user => {

            if (!user) {

                return res.status(404).send({ message: "Пользователь с таким логином не найден!" });

            }

            var passwordIsValid = bcrypt.compareSync(

                req.body.password,

                user.password

            );

            if (!passwordIsValid) {

                return res.status(401).send({

                    accessToken: null,

                    message: "Неверный пароль!"

                });

            }

            const token = jwt.sign({ id: user.id },

                config.secret,

                {

                    algorithm: 'HS256',

                    allowInsecureKeySizes: true,

                    expiresIn: 86400, // 24 hours

                });

            var authorities = [];

            user.getRoles().then(roles => {

                for (let i = 0; i < roles.length; i++) {

                    authorities.push("ROLE\_" + roles[i].name.toUpperCase());

                }

                res.status(200).send({

                    id: user.id,

                    username: user.username,

                    email: user.email,

                    name: user.name,

                    phone: user.phone,

                    birthdate: user.birthdate,

                    roles: authorities,

                    accessToken: token

                });

            })

        })

        .catch(err => {

            res.status(500).send({ message: err.message });

        });

};

**Приложение 6**

**Код в файле «App.js»**

<template>

<div *id*="app">

<div

:*class*="!showFooter ? 'background-image background-flex' : ''"

*class*="wrapper"

>

<*HeaderBlock* :*isShowBanner*="showBanner" />

<main>

<router-view />

</main>

<*FooterBlock* :*isShowFooter*="showFooter" />

</div>

</div>

</template>

<script>

import HeaderBlock from "[@/blocks/HeaderBlock.vue](mailto:@/blocks/HeaderBlock.vue)";

import FooterBlock from "[@/blocks/FooterBlock.vue](mailto:@/blocks/FooterBlock.vue)";

export default {

name: "App",

components: {

HeaderBlock,

FooterBlock,

},

computed: {

showBanner() {

const currentPage = *this*.$route.matched[0].components.default.name;

if (currentPage === "MainPage") return true;

return false;

},

showFooter() {

const currentPage = *this*.$route.matched[0].components.default.name;

switch (currentPage) {

case "MainPage":

return true;

case "AboutMaster":

return true;

case "ContactsPage":

return true;

case "MastersPage":

return true;

case "PricePage":

return true;

case "PromotionsPage":

return true;

default:

return false;

}

},

},

};

</script>

<style *lang*="scss">

html,

body {

height: 100%;

}

*#app* {

font-family: $font-main, sans-serif;

font-style: normal;

font-weight: 400;

color: $color-text;

background-color: $color-main;

}

\* {

margin: 0;

padding: 0;

}

*.container* {

max-width: $width-site;

margin: 0 auto;

}

*.background-flex* {

display: flex;

flex-direction: column;

justify-content: space-between;

}

*.background-image* {

min-height: 100vh;

background: url(*../src/assets/img/header-bg.jpg*) no-repeat center center;

background-size: cover;

}

*.wrapper* {

display: flex;

flex-direction: column;

min-height: 100vh;

}

main {

flex: 1 0 auto;

}

</style>

**Приложение 7**

**Код компонента «HeaderBlock.vue»**

<template>

<header *class*="header" :*class*="isShowBanner ? 'background-image' : ''">

<div *class*="border-bottom">

<nav *class*="header-menu container">

<a *href*="#" *class*="menu-link"

><i *class*="fa-brands fa-instagram soc-icon head-icon"></i>

</a>

<li *class*="menu-item">

<router-link *to*="/main" *class*="menu-link menu-border"

>Главная</router-link

>

</li>

<li *class*="menu-item">

<router-link

*to*="/masters"

*class*="menu-link menu-border"

:*class*="currentPath === '/masters' ? 'menu-active' : ''"

>Мастера</router-link

>

</li>

<li *class*="menu-item">

<router-link

*to*="/price"

*class*="menu-link menu-border"

:*class*="currentPath === '/price' ? 'menu-active' : ''"

>Стоимость</router-link

>

</li>

<li *class*="menu-item">

<router-link *to*="/main" *class*="menu-link"

><i *class*="fa-solid fa-signature head-icon"></i

></router-link>

</li>

<li *class*="menu-item">

<router-link

*to*="/promo"

*class*="menu-link menu-border"

:*class*="currentPath === '/promo' ? 'menu-active' : ''"

>Акции</router-link

>

</li>

<li *class*="menu-item">

<router-link

*to*="/shedule"

*class*="menu-link menu-border"

:*class*="currentPath === '/shedule' ? 'menu-active' : ''"

>Записаться</router-link

>

</li>

<li v-if="showAdminBoard" *class*="menu-item">

<router-link

*to*="/admin"

*class*="menu-link menu-border"

:*class*="currentPath === '/admin' ? 'menu-active' : ''"

>Администратор</router-link

>

</li>

<li v-if="showMasterBoard" *class*="menu-item">

<router-link

*to*="/master"

*class*="menu-link menu-border"

:*class*="currentPath === '/master' ? 'menu-active' : ''"

>Записи ко мне</router-link

>

</li>

<li v-if="!showAdminBoard && !showMasterBoard" *class*="menu-item">

<router-link

*to*="/contacts"

*class*="menu-link menu-border"

:*class*="currentPath === '/contacts' ? 'menu-active' : ''"

>Контакты</router-link

>

</li>

<li v-if="currentUser" *class*="menu-item">

<router-link

*to*="/profile"

*class*="menu-link"

:*class*="currentPath === '/profile' ? 'menu-active' : ''"

>

<i *class*="fa-solid fa-user head-icon"></i>

</router-link>

</li>

<li v-if="!currentUser" *class*="menu-item">

<router-link

*to*="/login"

*class*="menu-link"

:*class*="currentPath === '/login' ? 'menu-active' : ''"

>

<i *class*="fa-solid fa-user-plus head-icon"></i>

</router-link>

</li>

</nav>

</div>

<h1 *v-show*="isShowBanner" *class*="header\_\_title">

Салон красоты "Штаб бессовестно красивых"

</h1>

</header>

</template>

<script>

export default {

name: "HeaderBlock",

props: ["isShowBanner"],

computed: {

currentUser() {

return *this*.$store.state.auth.user;

},

showAdminBoard() {

if (*this*.currentUser && *this*.currentUser.roles) {

return *this*.currentUser.roles.includes("ROLE\_ADMIN");

}

return false;

},

showMasterBoard() {

if (*this*.currentUser && *this*.currentUser.roles) {

return *this*.currentUser.roles.includes("ROLE\_MASTER");

}

return false;

},

currentPath() {

const currentMenu = *this*.$route.path.split("/");

return `/${currentMenu[1]}`;

},

},

};

</script>

<style *lang*="scss" *scoped*>

*.border-bottom* {

border-bottom: 1px solid $color-border;

}

*.background-image* {

min-height: 100vh;

background: [url(](mailto:url(@/assets/img/header-bg.jpg)*[@/assets/img/header-bg.jpg](mailto:url(@/assets/img/header-bg.jpg)*) no-repeat center center;

background-size: cover;

}

*.header-menu* {

display: flex;

justify-content: space-between;

align-items: center;

padding: 30px;

box-sizing: border-box;

}

*.menu* {

&*-item* {

list-style-type: none;

}

&*-link* {

text-decoration: none;

color: $color-text;

font-weight: 600;

font-size: 22px;

line-height: 23px;

border: 1px solid transparent;

transition: all 0.3s;

}

&*-active* {

color: $color-hover;

}

&*-link:hover* {

color: $color-hover;

}

&*-border:hover.menu-link* {

border-bottom: 1px solid $color-hover;

}

}

*.head-icon* {

font-size: 40px;

line-height: 40px;

}

*.header\_\_title* {

font-weight: 700;

font-size: 72px;

line-height: 100%;

max-width: 600px;

text-align: center;

margin: 15vh auto;

}

</style>

**Приложение 8**

**Код в файле «router»/«index.js»**

import Vue from 'vue'

import VueRouter from 'vue-router'

import store from '../store/index.js'

import MainPage from '../views/MainPage.vue'

import MastersPage from '../views/MastersPage.vue'

import AboutMaster from '../views/AboutMaster.vue'

import NotFoundPage from '../views/NotFoundPage.vue'

import PricePage from '../views/PricePage.vue'

import PromotionsPage from '../views/PromotionsPage.vue'

import AppointmentPage from '../views/AppointmentPages/AppointmentPage.vue';

import AppointmentMaster from '../views/AppointmentPages/AppointmentMaster.vue';

import AppointmentService from '../views/AppointmentPages/AppointmentService.vue';

import AppointmentServiceGroup from '../views/AppointmentPages/AppointmentServiceGroup.vue';

import AppointmentMasterServ from '../views/AppointmentPages/AppointmentMasterServ.vue';

import MakeAppointment from '../views/AppointmentPages/MakeAppointment.vue';

import SuccessfulAppointment from '../views/AppointmentPages/SuccessfulAppointment.vue';

import ContactsPage from '../views/ContactsPage.vue'

import AccountPage from '../views/AccountPages/AccountPage.vue';

import LoginPage from '../views/AccountPages/LoginPage.vue';

import RegisterPage from '../views/AccountPages/RegisterPage.vue';

import AdminBoard from '../views/AdminBoard/AdminBoard.vue';

import SheduleUser from '../views/AccountPages/SheduleUser.vue';

import MasterBoard from '../views/MasterBoard/MasterBoard.vue';

import AdminUsers from '../views/AdminBoard/AdminUsers.vue';

import AdminMasters from '../views/AdminBoard/AdminMasters.vue';

import AdminServices from '../views/AdminBoard/AdminServices.vue';

import AdminShedules from '../views/AdminBoard/AdminShedules.vue';

Vue.use(VueRouter)

const routes = [

{

path: '/',

component: MainPage

},

{

path: '/main',

component: MainPage

},

{

path: '/masters',

component: MastersPage

},

{

path: '/masters/:masterID',

component: AboutMaster

},

{

path: '/price',

component: PricePage

},

{

path: '/promo',

component: PromotionsPage

},

{

path: '/shedule',

component: AppointmentPage,

meta: {

requiresAuth: true

}

},

{

path: '/shedule/master',

component: AppointmentMaster,

meta: {

requiresAuth: true

}

},

{

path: '/shedule/master/:idMaster',

component: AppointmentMasterServ,

meta: {

requiresAuth: true

}

},

{

path: '/shedule/master/:idMaster/:idService',

component: MakeAppointment,

meta: {

requiresAuth: true

}

},

{

path: '/shedule/service',

component: AppointmentService,

meta: {

requiresAuth: true

}

},

{

path: '/shedule/service/:idService',

component: MakeAppointment,

meta: {

requiresAuth: true

}

},

{

path: '/shedule/serviceGroup/:idGroup',

component: AppointmentServiceGroup,

meta: {

requiresAuth: true

}

},

{

path: '/shedule/master/:idMaster/:idService/:recDate',

component: SuccessfulAppointment,

meta: {

requiresAuth: true

}

},

{

path: '/contacts',

component: ContactsPage

},

{

path: '/login',

component: LoginPage

},

{

path: '/register',

component: RegisterPage

},

{

path: '/profile',

component: AccountPage

},

{

path: '/admin',

component: AdminBoard,

meta: {

requiresAuth: true

}

},

{

path: '/master',

component: MasterBoard,

meta: {

requiresAuth: true

}

},

{

path: '/userShedule',

component: SheduleUser,

meta: {

requiresAuth: true

}

},

{

path: '/admin/usersList',

component: AdminUsers,

meta: {

requiresAuth: true

}

},

{

path: '/admin/mastersList',

component: AdminMasters,

meta: {

requiresAuth: true

}

},

{

path: '/admin/servicesList',

component: AdminServices,

meta: {

requiresAuth: true

}

},

{

path: '/admin/shedulesList',

component: AdminShedules,

meta: {

requiresAuth: true

}

},

{

path: '\*',

component: NotFoundPage

}

]

const router = **new** *VueRouter*({

mode: 'history',

base: *process*.*env*.BASE\_URL,

routes

})

router.beforeEach((to, from, next) => {

if (to.matched.some(record => record.meta.requiresAuth)) {

if (store.state.auth.user) {

next()

return

}

next('/login')

} else {

next()

}

})

export default router

**Приложение 9**

**Код компонента «FooterBlock.vue»**

<template>

<footer *class*="footer">

<div *class*="border-top">

<div *v-show*="isShowFooter" *class*="container footer-box">

<router-link *to*="/main" *class*="footer-logo"

><i *class*="fa-solid fa-signature foot-logo"></i

></router-link>

<div *class*="contacts">

<h3 *class*="footer\_\_title">Контакты</h3>

<ul *class*="footer\_\_list">

<li>+7 (812) 123-45-67</li>

<li>+7 (911) 123-45-67</li>

<li>Новоостровский проспект, дом 36</li>

</ul>

</div>

<div *class*="regime">

<h3 *class*="footer\_\_title">Режим работы</h3>

<ul *class*="footer\_\_list">

<li>C 10:00 до 21:00 (Пн-Пт)</li>

<li>С 11:00 до 20:00 (Сб-Вс)</li>

</ul>

</div>

<div *class*="soc">

<h3 *class*="footer\_\_title">Мы в социальных сетях</h3>

<ul *class*="footer\_\_list footer-soc">

<li>

<a *href*="#" *class*="footer\_\_link"

><i *class*="fa-brands fa-instagram soc-icon footer-icon"></i

></a>

</li>

<li>

<a *href*="#" *class*="footer\_\_link"

><i *class*="fa-brands fa-vk footer-icon"></i

></a>

</li>

<li>

<a *href*="#" *class*="footer\_\_link"

><i *class*="fa-brands fa-twitter soc-icon footer-icon"></i

></a>

</li>

<li>

<a *href*="#" *class*="footer\_\_link"

><i *class*="fa-brands fa-telegram footer-icon"></i

></a>

</li>

</ul>

</div>

</div>

</div>

<div *class*="border-top">

<div *class*="container footer-copy">Copyright © 2017 - 2024</div>

</div>

</footer>

</template>

<script>

export default {

name: "FooterBlock",

props: ["isShowFooter"],

</script>

<style *lang*="scss" *scoped*>

*.border-top* {

border-top: 1px solid $color-border;

}

*.footer* {

&*-box* {

display: flex;

flex-wrap: wrap;

justify-content: space-between;

padding: 20px 10px;

box-sizing: border-box;

}

&*-logo* {

display: block;

text-decoration: none;

}

&*\_\_title* {

font-weight: 700;

font-size: 24px;

line-height: normal;

margin-bottom: 30px;

}

&*\_\_list* {

list-style: none;

}

&*\_\_list* > li {

font-size: 18px;

line-height: normal;

margin-bottom: 18px;

}

&*-soc* {

display: flex;

flex-wrap: wrap;

justify-content: space-around;

}

&*-icon* {

font-size: 36px;

line-height: 36px;

color: $color-text;

&*:hover* {

color: $color-hover;

}

}

&*-copy* {

font-weight: 700;

font-size: 18px;

line-height: normal;

text-align: center;

padding: 20px;

}

}

*.foot-logo* {

color: $color-text;

font-size: 65px;

line-height: 65px;

transition: all 0.3s;

&*:hover* {

color: $color-hover;

}

}

</style>

**Приложение 10**

**Код компонента «MainPage.vue»**

<template>

<div>

<*QuoteComponent* />

<*ServicesBlock* />

<*WorksBlock* />

</div>

</template>

<script>

import QuoteComponent from "[@/components/QuoteComponent.vue](mailto:@/components/QuoteComponent.vue)";

import ServicesBlock from "[@/blocks/ServicesBlock.vue](mailto:@/blocks/ServicesBlock.vue)";

import WorksBlock from "[@/blocks/WorksBlock.vue](mailto:@/blocks/WorksBlock.vue)";

export default {

name: "MainPage",

components: {

QuoteComponent,

ServicesBlock,

WorksBlock,

},

};

</script>

<style *lang*="scss" *scoped*></style>

**Приложение 11**

**Код компонента «QuoteComponent.vue»**

<template>

<section *class*="quote">

<div *class*="container quote\_\_box">

<img *src*="../assets/img/quote.png" *alt*="quote" />

<blockquote *class*="quote\_\_text">

<p>

{{ GET\_RANDOM\_QUOT.text }}

</p>

<cite *class*="quote\_\_author"> {{ GET\_RANDOM\_QUOT.author }}</cite>

</blockquote>

</div>

</section>

</template>

<script>

import { mapGetters } from "vuex";

export default {

name: "QuoteComponent",

computed: {

...mapGetters(["GET\_RANDOM\_QUOT"]),

},

};

</script>

<style *lang*="scss" *scoped*>

*.quote* {

margin-bottom: 95px;

margin-top: 120px;

&*\_\_box* {

max-width: 856px;

display: flex;

flex-direction: column;

align-items: center;

gap: 60px;

}

&*\_\_text* {

font-weight: 700;

font-size: 27px;

line-height: 130%;

text-align: center;

display: flex;

flex-direction: column;

gap: 30px;

}

&*\_\_author* {

align-self: end;

color: $color-hover;

}

}

</style>

**Приложение 12**

**Код хранилища «store»/«index.js»**

import Vue from 'vue'

import Vuex from 'vuex'

import { auth } from './auth.module';

import { masterStore } from './master.module';

import { servicesStore } from './services.module';

import { sheduleStore } from './shedule.module';

import { userStore } from './user.module';

Vue.use(Vuex)

export default new Vuex.Store({

  state: {

    quotes: [

      { author: 'Софи Лорен', text: 'Прическа влияет на то, как складывается день, а в итоге и жизнь.' },

      { author: 'Янина Ипохорская', text: 'Даже самую лучшую прическу нужно иногда менять' },

      { author: 'Одри Хепберн', text: 'Я верю в маникюр, в кричащую одежду, в то, что на отдыхе тоже необходимо делать прическу и наносить губную помаду' },

      { author: 'Эвелина Хромченко', text: 'Макияж - это одежда для лица. Не грешите, выходя на улицу голыми' },

      { author: 'Кэльвин Кляйн', text: 'Самое главное в макияже - выглядеть совершенно естественно; но для этого нужно очень много косметики' },

    ],

    groupServices: [

      { id: 1, title: 'Парикмахерские услуги', image: 'serv1.jpg' },

      { id: 2, title: 'Маникюр', image: 'serv2.jpg' },

      { id: 3, title: 'Педикюр', image: 'serv3.jpg' },

      { id: 4, title: 'Косметология', image: 'serv4.jpg' },

      { id: 5, title: 'Эстетист по телу', image: 'serv5.jpg' },

      { id: 6, title: 'Визаж', image: 'serv6.jpg' },

    ],

    worksImage: [

      { id: 1, img: 'work-1.jpg', tagID: 1 },

      { id: 2, img: 'work-2.jpg', tagID: 3 },

      { id: 3, img: 'work-3.jpg', tagID: 2 },

      { id: 4, img: 'work-4.jpg', tagID: 1 },

      { id: 5, img: 'work-5.jpg', tagID: 1 },

      { id: 6, img: 'work-6.jpg', tagID: 3 },

      { id: 7, img: 'work-7.jpg', tagID: 2 },

      { id: 8, img: 'work-8.jpg', tagID: 3 },

      { id: 9, img: 'work-9.jpg', tagID: 3 }

    ],

    actions: [

      { id: 1, text: "Первое посещение - скидка 20%", image: "actions1.jpg" },

      { id: 2, text: "Каждая пятая стрижка бесплатно", image: "actions2.jpg" },

    ],

  },

  getters: {

    GET\_RANDOM\_QUOT(state) {

      const index = Math.floor(Math.random() \* state.quotes.length);

      return state.quotes[index];

    },

    GROUP\_SERVICES(state) {

      return state.groupServices;

    },

    GET\_GROUP\_SERVICE\_FOR\_ID: state => (id) => state.groupServices.find(el => el.id === id),

    GET\_WORKSIMAGE(state) {

      return state.worksImage;

    },

    GET\_ACTIONS(state) {

      return state.actions;

    },

  },

  mutations: {

  },

  actions: {

  },

  modules: {

    auth,

    masterStore,

    servicesStore,

    sheduleStore,

    userStore

  }

})

**Приложение 13**

**Код компонента «ServicesBlock.vue»**

<template>

<section *class*="services">

<div *class*="container services\_\_box">

<router-link

v-for="item in GROUP\_SERVICES"

:*key*="item.id"

:*to*="`shedule/serviceGroup/${item.id}`"

*class*="services\_\_item"

>

<img [:](mailto::src=require(`@/assets/img/${item.image}`)*[src](mailto::src=require(`@/assets/img/${item.image}`)*[="require(`@/assets/img/${item.image}`](mailto::src=require(`@/assets/img/${item.image}`))" *alt*="our\_services" />

<h3 *class*="services\_\_title">{{ item.title }}</h3>

</router-link>

</div>

<div *class*="container materials-box">

<img *[src](mailto:src=@/assets/img/mater1.png)*[="@/assets/img/mater1.png](mailto:src=@/assets/img/mater1.png)" *alt*="logo" *class*="materials-img" />

<img *[src](mailto:src=@/assets/img/mater2.png)*[="@/assets/img/mater2.png](mailto:src=@/assets/img/mater2.png)" *alt*="logo" *class*="materials-img" />

<img *[src](mailto:src=@/assets/img/mater3.png)*[="@/assets/img/mater3.png](mailto:src=@/assets/img/mater3.png)" *alt*="logo" *class*="materials-img" />

<img *[src](mailto:src=@/assets/img/mater4.png)*[="@/assets/img/mater4.png](mailto:src=@/assets/img/mater4.png)" *alt*="logo" *class*="materials-img" />

</div>

</section>

</template>

<script>

import { mapGetters } from "vuex";

export default {

name: "ServicesBlock",

computed: {

...mapGetters(["GROUP\_SERVICES"]),

},

};

</script>

<style *lang*="scss" *scoped*>

*.services* {

&*\_\_box* {

display: flex;

flex-wrap: wrap;

justify-content: center;

gap: 44px;

margin-bottom: 150px;

}

&*\_\_item* {

margin-bottom: 36px;

position: relative;

text-decoration: none;

color: $color-text;

z-index: 10;

}

&*\_\_title::before* {

content: "";

display: block;

width: 396px;

height: 300px;

border: 1px solid transparent;

position: absolute;

top: 0px;

left: 0px;

z-index: -10;

transition: all 0.3s;

}

&*\_\_title* {

font-size: 30px;

line-height: 120%;

margin-top: 30px;

font-weight: 400;

transition: all 0.3s;

}

&*\_\_item:hover* &*\_\_title* {

color: $color-hover;

}

&*\_\_item:hover* &*\_\_title::before* {

border: 1px solid $color-hover;

transform: translate(20px, -20px);

}

}

*.materials-box* {

display: flex;

justify-content: space-around;

align-items: center;

margin-bottom: 160px;

}

</style>

**Приложение 14**

**Код компонента «WorksBlock.vue»**

<template>

<section *class*="works container">

<h2 *class*="pages\_\_title">Наши работы</h2>

<ul *class*="works\_\_filter">

<li

v-for="tag in tags"

:*key*="tag.id"

@*click*="changeCurrentTagID(tag.id)"

*class*="filter-item"

:*class*="tag.id === currentTagID ? 'activeTag' : ''"

>

{{ tag.name }}

</li>

</ul>

<div *class*="works\_\_box">

<img

v-for="image in filterByTag"

:*key*="image.id"

[:](mailto::src=require(`@/assets/img/${image.img}`)*[src](mailto::src=require(`@/assets/img/${image.img}`)*[="require(`@/assets/img/${image.img}`](mailto::src=require(`@/assets/img/${image.img}`))"

*alt*="our works"

*class*="works\_\_foto"

/>

</div>

</section>

</template>

<script>

import { mapGetters } from "vuex";

export default {

name: "WorksBlock",

data() {

return {

tags: [

{ id: 0, name: "Показать все" },

{ id: 1, name: "Парикмахерские услуги" },

{ id: 2, name: "Маникюр" },

{ id: 3, name: "Визаж" },

],

currentTagID: 0,

};

},

computed: {

...mapGetters(["GET\_WORKSIMAGE"]),

filterByTag() {

if (*this*.currentTagID) {

return *this*.GET\_WORKSIMAGE.filter(

(*el*) => *el*.tagID === *this*.currentTagID

);

} else {

return *this*.GET\_WORKSIMAGE;

}

},

},

methods: {

changeCurrentTagID(*id*) {

*this*.currentTagID = *id*;

},

},

};

</script>

<style *lang*="scss" *scoped*>

*.works* {

margin-bottom: 100px;

}

*.pages\_\_title* {

font-weight: 700;

font-size: 48px;

line-height: 120%;

text-align: center;

margin-bottom: 80px;

}

*.works\_\_filter* {

list-style: none;

display: flex;

gap: 60px;

justify-content: center;

font-size: 24px;

line-height: 130%;

margin-bottom: 60px;

}

*.filter-item* {

border-bottom: 1px solid transparent;

transition: all 0.3s;

}

*.filter-item:hover* {

color: $color-hover;

border-bottom: 1px solid $color-hover;

cursor: pointer;

}

*.activeTag* {

color: $color-hover;

}

*.works\_\_box* {

display: flex;

flex-wrap: wrap;

gap: 24px;

justify-content: center;

}

</style>

**Приложение 15**

**Код компонента «MastersPage.vue»**

<template>

<section *class*="masters pages-margin">

<div *class*="container">

<h2 *class*="pages\_\_title">Наши мастера</h2>

<div *class*="masters\_\_box">

<router-link

:*to*="`/masters/${master.id}`"

v-for="master in MASTERS"

:*key*="master.id"

*class*="masters\_\_item"

>

<img

[:](mailto::src=require(`@/assets/img/${master.photo}`)*[src](mailto::src=require(`@/assets/img/${master.photo}`)*[="require(`@/assets/img/${master.photo}`](mailto::src=require(`@/assets/img/${master.photo}`))"

*alt*="our\_masters"

*class*="masters\_\_photo"

/>

<h3 *class*="masters\_\_name">{{ master.name }}</h3>

<p *class*="masters\_\_about">{{ master.position }}</p>

</router-link>

</div>

</div>

</section>

</template>

<script>

import { mapGetters } from "vuex";

export default {

name: "MastersPage",

computed: {

...mapGetters(["MASTERS"]),

},

mounted() {

*this*.$store.dispatch("GET\_MASTERS");

},

};

</script>

<style *lang*="scss" *scoped*>

*.pages-margin* {

margin-top: 80px;

margin-bottom: 100px;

}

*.masters* {

&*\_\_box* {

display: flex;

flex-wrap: wrap;

justify-content: center;

gap: 40px;

}

&*\_\_item* {

margin-bottom: 40px;

position: relative;

color: $color-text;

text-decoration: none;

&*:hover* *.masters\_\_name* {

color: $color-hover;

}

}

*.masters\_\_name::before* {

content: "";

display: block;

width: 400px;

height: 477px;

border: 1px solid $color-hover;

position: absolute;

top: -16px;

left: 16px;

}

&*\_\_name* {

font-weight: 700;

font-size: 27px;

line-height: 130%;

margin-top: 30px;

}

&*\_\_about* {

font-weight: 400;

font-size: 18px;

line-height: 130%;

margin-top: 10px;

}

&*\_\_photo* {

position: relative;

z-index: 10;

}

}

*.pages\_\_title* {

font-weight: 700;

font-size: 48px;

line-height: 120%;

text-align: center;

margin-bottom: 80px;

}

</style>

**Приложение 16**

**Код модуля хранилища «master.module.js»**

import Master from "../models/Master";

import masterService from "../services/master.service"  
export const masterStore = {

    state: {

        masters: [],

        currentMaster: null,

        masterAccount: null,

    },

    getters: {

        MASTERS(*state*) {

            return *state*.masters;

        },

        CURRENT\_MASTER(*state*) {

            return *state*.currentMaster;

        },

        MASTER\_ACCOUNT(*state*) {

            return *state*.masterAccount;

        },

        MASTER\_BY\_ID: *state* => *id* => {

            return *state*.masters.find(*master* => *master*.id === *id*);

        },

    },

    mutations: {

        SET\_MASTERS(*state*, *mastersList*) {

*state*.masters = *mastersList*;

        },

        ADD\_MASTER: (*state*, *master*) => {

*state*.masters.push(*master*);

        },

        SET\_CURRENT\_MASTER: (*state*, *master*) => {

*state*.currentMaster = *master*;

        },

        SET\_MASTER\_ACCOUNT: (*state*, *master*) => {

*state*.masterAccount = *master*;

        },

        DEL\_MASTER: (*state*, *id*) => {

*state*.masters.splice(*state*.masters.findIndex(*el* => *el*.id === *id*), 1);

        },

    },

    actions: {

        GET\_MASTERS: async (*context*) => {

            masterService.getAllMasters().then((*res*) => {

                const data = [];

*res*.data.forEach((*element*) => {

                    data.push(**new** Master(*element*.id, *element*.master\_name, *element*.birthdate, *element*.phone, *element*.position, *element*.photo, *element*.about\_text, *element*.group\_service\_id));

                });

*context*.commit('SET\_MASTERS', data);

            })

        },

        GET\_MASTERS\_BY\_GROUP: async (*context*, *idGroup*) => {

            masterService.getMastersByGroup(*idGroup*).then((*res*) => {

                const data = [];

*res*.data.forEach((*element*) => {

                    data.push(**new** Master(*element*.id, *element*.master\_name, *element*.birthdate, *element*.phone, *element*.position, *element*.photo, *element*.about\_text, *element*.group\_service\_id));

                });

*context*.commit('SET\_MASTERS', data);

            })

        },

        GET\_MASTER\_BY\_ID: async (*context*, *idMaster*) => {

            masterService.GetMasterById(*idMaster*).then((*res*) => {

                const data = **new** Master(*res*.data.id, *res*.data.master\_name, *res*.data.birthdate, *res*.data.phone, *res*.data.position, *res*.data.photo, *res*.data.about\_text, *res*.data.group\_service\_id)

*context*.commit('SET\_CURRENT\_MASTER', data);

            })

        },

        GET\_MASTER\_BY\_NAME\_AND\_PHONE: async (*context*, *params*) => {

            return **new** *Promise*((resolve) => {

                masterService.getMasterByNameAndPhone(params).then((res) => {

                    const master = **new** *Master*(res.data.id, res.data.master\_name, res.data.birthdate, res.data.phone, res.data.position, res.data.photo, res.data.about\_text, res.data.group\_service\_id)

                    context.commit('SET\_MASTER\_ACCOUNT', master);

                    resolve(master);

                })

            })

        },

        DELETE\_MASTER: async (*context*, *id*) => {

            masterService.deleteMasterById(*id*).then((*res*) => {

                console.log(*res*.data);

*context*.commit('DEL\_MASTER', *id*);

            })

                .catch((*e*) => {

                    console.log(*e*);

                });

        },

        CREATE\_MASTER: async (*context*, *data*) => {

            masterService.createMaster(*data*).then((*res*) => {

                console.log(*res*.data);

*context*.commit('ADD\_MASTER', **new** Master(*res*.data.id, *res*.data.master\_name, *res*.data.birthdate, *res*.data.phone, *res*.data.position, *res*.data.photo, *res*.data.about\_text, *res*.data.group\_service\_id));

            })

                .catch((*e*) => {

                    console.log(*e*);

                });

        },

        UPDATE\_MASTER: async (*context*, *data*) => {

            return **new** *Promise*((resolve) => {

                masterService.updateMaster(*data*).then((*res*) => {

                    console.log(*res*);

                    resolve(*res*)

                })

            }).catch((*e*) => {

                console.log(*e*);

            });

        },

    },

}

**Приложение 17**

**Код сервиса «master.service.js»**

import axios from 'axios';

const API\_URL = "http://localhost:8080/api/masters";

class MasterService {

    getAllMasters() {

        return axios.get(API\_URL);

    }

    getMastersByGroup(*idGroup*) {

        return axios.get(API\_URL + `?group\_service\_id=${*idGroup*}`);

    }

    GetMasterById(*idMaster*) {

        return axios.get(API\_URL + `/${*idMaster*}`);

    }

    getMasterByNameAndPhone(*params*) {

        return axios.get(API\_URL + `/masterlk/master?name=${*params*.name}&phone=${*params*.phone}`);

    }

    deleteMasterById(*idMaster*) {

        return axios.delete(API\_URL + `/${*idMaster*}`);

    }

    createMaster(*data*){

        return axios.post(API\_URL, *data*);

    }

    updateMaster(*data*){

        return axios.put(API\_URL + `/${*data*.id}`, *data*.master );

    }

}

export default new MasterService();

**Приложение 18**

**Код компонента «AboutMaster.vue»**

<template>

<section *class*="container">

<div *class*="aboutMaster">

<div *class*="photo">

<img

[:](mailto::src=require(`@/assets/img/${CURRENT_MASTER.photo}`)*[src](mailto::src=require(`@/assets/img/${CURRENT_MASTER.photo}`)*[="require(`@/assets/img/${CURRENT\_MASTER.photo}`](mailto::src=require(`@/assets/img/${CURRENT_MASTER.photo}`))"

*alt*="our\_master"

/>

</div>

<div *class*="aboutMaster\_\_content">

<p *class*="aboutMaster\_\_text">{{ CURRENT\_MASTER.name }}</p>

<p *class*="aboutMaster\_\_text">{{ CURRENT\_MASTER.position }}</p>

<p *class*="aboutMaster\_\_text">

{{ CURRENT\_MASTER.aboutText }}

</p>

<router-link

:*to*="`/shedule/master/${CURRENT\_MASTER.id}`"

*class*="subscribeBtn"

>Записаться к мастеру</router-link

>

</div>

</div>

</section>

</template>

<script>

import { mapGetters } from "vuex";

export default {

name: "AboutMaster",

computed: {

...mapGetters(["CURRENT\_MASTER"]),

},

mounted() {

const masterId = +*this*.$route.params.masterID;

*this*.$store.dispatch("GET\_MASTER\_BY\_ID", masterId);

},

};

</script>

<style *lang*="scss" *scoped*>

*.aboutMaster* {

display: flex;

gap: 100px;

margin-bottom: 150px;

margin-top: 150px;

&*\_\_content* {

font-size: 24px;

line-height: 130%;

}

&*\_\_text* {

margin-bottom: 50px;

}

&*\_\_text:first-of-type* {

font-weight: 700;

font-size: 48px;

line-height: 120%;

margin-bottom: 0px;

}

}

*.subscribeBtn* {

height: 60px;

background-color: #cdaa7d;

border-radius: 10px;

color: #0a1111;

border: 1px solid transparent;

cursor: pointer;

font-family: "Cormorant Garamond";

font-size: 24px;

font-style: normal;

font-weight: 700;

line-height: normal;

box-sizing: border-box;

text-decoration: none;

padding: 10px 30px;

transition: all 0.3s;

&*:hover* {

background-color: white;

border: 1px solid #cdaa7d;

}

}

</style>

**Приложение 19**

**Код компонента «PricePage.vue»**

<template>

<section *class*="price pages-margin">

<div *class*="container">

<h2 *class*="pages\_\_title">Цены на услуги</h2>

<div *class*="price\_\_box">

<div *class*="price\_\_item" v-for="idGroupService in idOfGroupServices" :*key*="idGroupService">

<img

*class*="price\_\_img"

[:](mailto::src=require(`@/assets/img/price${idGroupService}.jpg`)*[src](mailto::src=require(`@/assets/img/price${idGroupService}.jpg`)*[="require(`@/assets/img/price${idGroupService}.jpg`](mailto::src=require(`@/assets/img/price${idGroupService}.jpg`))"

*alt*="foto"

/>

<table *class*="price-table">

<div

v-for="service in servicesByGroupService(idGroupService)"

:*key*="service.id"

*class*="price\_\_row"

>

<tr *class*="price\_\_row-first">

<td *class*="bigger-row">{{ service.title }}</td>

<td *class*="price\_\_value">{{ service.price }} руб</td>

</tr>

<tr>

<td *colspan*="2" *class*="smaller-row">

{{ titleOfGroupService(idGroupService) }}

</td>

</tr>

</div>

</table>

</div>

</div>

</div>

</section>

</template>

<script>

import { mapGetters } from "vuex";

export default {

name: "PricePage",

computed: {

...mapGetters(["SERVICES", "GROUP\_SERVICES"]),

// для группировки услуг в прайсе по группам

idOfGroupServices() {

const result = [];

*this*.SERVICES.forEach((*element*) => {

result.push(*element*.groupServiceId);

});

return [...new *Set*(result)];

},

},

methods: {

servicesByGroupService(*groupServiceId*){

return *this*.SERVICES.filter(

(*service*) => *service*.groupServiceId === *groupServiceId*

)

},

titleOfGroupService(*idGroupService*){

return *this*.GROUP\_SERVICES.find((*el*) => *el*.id === *idGroupService*).title

}

},

mounted() {

*this*.$store.dispatch('GET\_SERVICES');

},

};

</script>

<style *lang*="scss" *scoped*>

*.price* {

&*\_\_row* {

border-bottom: 1px solid $color-text;

}

&*\_\_row-first* {

display: flex;

justify-content: space-between;

}

}

*.price\_\_box* {

display: flex;

flex-direction: column;

gap: 80px;

}

*.price\_\_item* {

display: flex;

justify-content: space-between;

align-items: center;

gap: 134px;

}

*.price\_\_item:nth-child*(2n) {

flex-direction: row-reverse;

}

*.price\_\_img* {

display: block;

width: 100%;

}

*.price-table* {

font-weight: 700;

font-size: 27px;

line-height: 130%;

width: 100%;

border-collapse: collapse;

}

*.bigger-row* {

padding-bottom: 10px;

padding-top: 30px;

}

*.price\_\_value* {

text-align: right;

padding-bottom: 10px;

padding-top: 30px;

}

*.smaller-row* {

font-weight: 400;

font-size: 18px;

padding-bottom: 10px;

}

*.pages-margin* {

margin-top: 80px;

margin-bottom: 100px;

}

*.pages\_\_title* {

font-weight: 700;

font-size: 48px;

line-height: 120%;

text-align: center;

margin-bottom: 80px;

}

</style>

**Приложение 20**

**Код модуля хранилища «service.module.js»**

import Service from "../models/Service";

import servicesService from "../services/services.service";

export const servicesStore = {

  state: {

    services: [],

    currentService: null,

    serviceById: null,

  },

  getters: {

    SERVICES(state) {

      return state.services;

    },

    CURRENT\_SERVICE(state) {

      return state.currentService;

    },

    SERVICE\_BY\_ID(state) {

      return state.serviceById;

    },

    SERVICE\_FOR\_ID: (state) => (id) => {

      return state.services.find((el) => el.id === id);

    },

  },

  mutations: {

    SET\_SERVICES(state, servicesList) {

      state.services = servicesList;

    },

    ADD\_SERVICE: (state, service) => {

      state.services.push(service);

    },

    SET\_CURRENT\_SERVICE: (state, service) => {

      state.currentService = service;

    },

    SET\_SERVICE\_BY\_ID: (state, service) => {

      state.serviceById = service;

    },

    DEL\_SERVICE: (state, id) => {

      state.services.splice(

        state.services.findIndex((el) => el.id === id),

        1

      );

    },

  },

  actions: {

    GET\_SERVICES: async (context) => {

      servicesService

        .getAllServices()

        .then((res) => {

          const data = [];

          res.data.forEach((element) => {

            data.push(

**new** *Service*(

                element.id,

                element.title,

                element.group\_service\_id,

                element.duration,

                element.price

              )

            );

          });

          context.commit("SET\_SERVICES", data);

        })

        .catch((e) => {

*console*.log(e);

        });

    },

    GET\_SERVICES\_BY\_GROUP: async (context, id) => {

      servicesService

        .getServicesByGroupId(id)

        .then((res) => {

          const data = [];

          res.data.forEach((element) => {

            data.push(

**new** *Service*(

                element.id,

                element.title,

                element.group\_service\_id,

                element.duration,

                element.price

              )

            );

          });

          return *context*.commit("SET\_SERVICES", data);

        })

        .catch((*e*) => {

          console.log(*e*);

        });

    },

    GET\_SERVICE\_BY\_ID: async (*context*, *id*) => {

      servicesService

        .getServiceById(*id*)

        .then((*res*) => {

          const data = **new** Service(

*res*.data.id,

*res*.data.title,

*res*.data.group\_service\_id,

*res*.data.duration,

*res*.data.price

          );

*context*.commit("SET\_SERVICE\_BY\_ID", data);

        })

        .catch((*e*) => {

          console.log(*e*);

        });

    },

    CREATE\_SERVICE: async (*context*, *data*) => {

      servicesService

        .createService(*data*)

        .then((*res*) => {

*context*.commit(

            "ADD\_SERVICE",

**new** Service(

*res*.data.id,

*res*.data.title,

*res*.data.group\_service\_id,

*res*.data.duration,

*res*.data.price

            )

          );

        })

        .catch((*e*) => {

          console.log(*e*);

        });

    },

    UPDATE\_SERVICE: async (*context*, *data*) => {

      return **new** *Promise*((resolve) => {

        servicesService.updateService(*data*).then((*res*) => {

          console.log(*res*);

          resolve(*res*);

        });

      }).catch((*e*) => {

        console.log(*e*);

      });

    },

    DELETE\_SERVICE: async (*context*, *id*) => {

      servicesService

        .deleteServiceById(*id*)

        .then((*res*) => {

          console.log(*res*.data);

*context*.commit("DEL\_SERVICE", *id*);

        })

        .catch((*e*) => {

          console.log(*e*);

        });

    },

  },

};

**Приложение 21**

**Код сервиса «services.service.js»**

import axios from 'axios';

const API\_URL = "http://localhost:8080/api/services";

class ServicesService {

    getAllServices() {

        return axios.get(API\_URL);

    }

    getServicesByGroupId(*id*) {

        return axios.get(API\_URL + `/byGroupService?group\_service\_id=${*id*}`);

    }

    getServiceById(*id*){

        return axios.get(API\_URL + `/servicesById/${*id*}`);

    }

    createService(*data*){

        return axios.post(API\_URL, *data*);

    }

    updateService(*data*){

        return axios.put(API\_URL + `/${*data*.id}`, *data*.service );

    }

    deleteServiceById(*id*) {

        return axios.delete(API\_URL + `/${*id*}`);

    }

}

export default new ServicesService();

**Приложение 22**

**Код компонента «PromotionsPage.vue»**

<template>

<main *class*="actions pages-margin">

<div *class*="container">

<h2 *class*="pages\_\_title">Наши акции</h2>

<*ActionComponent* v-for="action in GET\_ACTIONS" :*key*="action.id" :*action*="action"/>

</div>

</main>

</template>

<script>

import { mapGetters } from "vuex";

import ActionComponent from "[@/components/ActionComponent.vue](mailto:@/components/ActionComponent.vue)";

export default {

name: "PromotionsPage",

components: { ActionComponent },

computed: {

...mapGetters(["GET\_ACTIONS"]),

}

};

</script>

<style *lang*="scss" *scoped*>

*.pages\_\_title* {

font-weight: 700;

font-size: 48px;

line-height: 120%;

text-align: center;

margin-top: 40px;

margin-bottom: 80px;

}

</style>

**Приложение 23**

**Код компонента «ActionComponent.vue»**

<template>

<div *class*="action-banner">

<img

*class*="action-img"

[:](mailto::src=require(`@/assets/img/${action.image}`)*[src](mailto::src=require(`@/assets/img/${action.image}`)*[="require(`@/assets/img/${action.image}`](mailto::src=require(`@/assets/img/${action.image}`))"

*alt*="action"

/>

<p *class*="action-text">{{ action.text }}</p>

</div>

</template>

<script>

export default {

name: "ActionComponent",

props: ["action"],

};

</script>

<style *lang*="scss" *scoped*>

*.pages\_\_title* {

font-weight: 700;

font-size: 48px;

line-height: 120%;

text-align: center;

margin-top: 40px;

margin-bottom: 80px;

}

*.action-banner* {

max-height: 60vh;

background-size: cover;

margin-bottom: 100px;

font-size: 70px;

box-sizing: border-box;

text-align: center;

position: relative;

overflow: hidden;

}

*.action-img* {

width: 100%;

}

*.action-text* {

position: absolute;

bottom: 5%;

}

</style>

**Приложение 24**

**Код компонента «ContactsPage.vue»**

<template>

<main *class*="contacts pages-margin">

<div *class*="container">

<h2 *class*="pages\_\_title">Наши контакты</h2>

<div *class*="contacts\_\_map">

<a

*class*="contacts\_\_link"

*href*="https://yandex.ru/maps/um=constructor%3A2ce4631c20907488940aef6b08252d33b512861279e44f5b4fd6a92961273edd&amp;source=constructorStatic"

*target*="\_blank"

><img

*class*="contacts\_\_img"

*src*="https://api-maps.yandex.ru/services/constructor/1.0/static/?um=constructor%3A2ce4631c20907488940aef6b08252d33b512861279e44f5b4fd6a92961273edd&amp;width=500&amp;height=400&amp;lang=ru\_RU"

*alt*=""

*style*="border: 0"

/></a>

</div>

<div *class*="contacts\_\_info">

<h3 *class*="contacts\_\_title">Контакты</h3>

<ul *class*="contacts\_\_content">

<li *class*="contacts\_\_item">+7 (812) 123-45-67</li>

<li *class*="contacts\_\_item">+7 (911) 123-45-67</li>

<li *class*="contacts\_\_item">Новоостровский проспект, дом 36</li>

</ul>

<h3 *class*="contacts\_\_title">Режим работы</h3>

<ul *class*="contacts\_\_content">

<li *class*="contacts\_\_item">C 10:00 до 21:00 (Пн-Пт)</li>

<li *class*="contacts\_\_item">С 11:00 до 20:00 (Сб-Вс)</li>

</ul>

</div>

</div>

</main>

</template>

<script>

export default {

name: "ContactsPage",

};

</script>

<style *lang*="scss" *scoped*>

*.contacts* {

&*\_\_map* {

filter: grayscale(50%);

display: flex;

}

&*\_\_link* {

display: block;

margin: 0 auto;

}

&*\_\_img* {

width: 100%;

}

&*\_\_info* {

display: flex;

justify-content: space-around;

margin-top: 80px;

}

&*\_\_title* {

font-weight: 700;

font-size: 34px;

line-height: 120%;

margin-bottom: 34px;

}

&*\_\_content* {

list-style: none;

font-size: 24px;

line-height: 130%;

}

&*\_\_item*{

margin-bottom: 15px;

}

}

*.pages\_\_title* {

font-weight: 700;

font-size: 48px;

line-height: 120%;

text-align: center;

margin-top: 30px;

margin-bottom: 30px;

}

</style>

**Приложение 25**

**Код сервисов аутентификации**

**Код в файле «auth.service.js»**

import axios from 'axios';

const API\_URL = '<http://localhost:8080/api/auth/>';

class AuthService {

login(*user*) {

return axios

.post(API\_URL + 'signin', {

username: *user*.username,

password: *user*.password

})

.then(*response* => {

if (*response*.data.accessToken) {

localStorage.setItem('user', JSON.stringify(*response*.data));

}

return *response*.data;

});

}

logout() {

localStorage.removeItem('user');

}

register(*user*) {

return axios.post(API\_URL + 'signup', {

username: *user*.username,

email: *user*.email,

password: *user*.password,

name: *user*.name,

phone: *user*.phone,

birthdate: *user*.birthdate

});

}

}

export default new AuthService();

**Код в файле «auth-header.js»**

export default function authHeader() {

let user = JSON.parse(localStorage.getItem('user'));

if (user && user.accessToken) {

return { 'x-access-token': user.accessToken };

} else {

return {};

}

}

**Приложение 26**

**Код сервиса «user.service.js»**

import axios from 'axios';

import authHeader from './auth-header';

const API\_URL = '<http://localhost:8080/api/>';

class UserService {

getAllUsers() {

return axios.get(API\_URL + 'test/all');

}

getUserBoard() {

return axios.get(API\_URL + 'test/user', { headers: authHeader() });

}

getMasterBoard() {

return axios.get(API\_URL + 'test/master', { headers: authHeader() });

}

getAdminBoard() {

return axios.get(API\_URL + 'test/admin', { headers: authHeader() });

}

getAllUsersForAdmin(*params*) {

if ('phone' in *params*) {

return axios.get(API\_URL + `users?start=${*params*.start}&count=${*params*.count}&phone=${*params*.phone}`, { headers: authHeader() });

} else {

return axios.get(API\_URL + `users?start=${*params*.start}&count=${*params*.count}`, { headers: authHeader() });

}

}

getUsersCount(*phone*) {

if(*phone*){

return axios.get(API\_URL + `usersCount?phone=${*phone*}`, { headers: authHeader() });

}else{

return axios.get(API\_URL + `usersCount`, { headers: authHeader() });

}

}

deleteUser(*id*) {

return axios.delete(API\_URL + `users/${*id*}`, { headers: authHeader() });

}

}

export default **new** UserService();

**Приложение 27**

**Код хранилища «auth.module.js»**

import AuthService from '../services/auth.service';

const user = JSON.parse(localStorage.getItem('user'));

const initialState = user

? { status: { loggedIn: true }, user }

: { status: { loggedIn: false }, user: null };

export const auth = {

namespaced: true,

state: initialState,

actions: {

login({ *commit* }, *user*) {

return AuthService.login(*user*).then(

*user* => {

commit('loginSuccess', *user*);

return *Promise*.resolve(*user*);

},

*error* => {

commit('loginFailure');

return *Promise*.reject(*error*);

}

);

},

logout({ *commit* }) {

AuthService.logout();

commit('logout');

},

register({ *commit* }, *user*) {

return AuthService.register(*user*).then(

*response* => {

commit('registerSuccess');

return *Promise*.resolve(*response*.data);

},

*error* => {

commit('registerFailure');

return *Promise*.reject(*error*);

}

);

}

},

mutations: {

loginSuccess(*state*, *user*) {

*state*.status.loggedIn = true;

*state*.user = *user*;

},

loginFailure(*state*) {

*state*.status.loggedIn = false;

*state*.user = null;

},

logout(*state*) {

*state*.status.loggedIn = false;

*state*.user = null;

},

registerSuccess(*state*) {

*state*.status.loggedIn = false;

},

registerFailure(*state*) {

*state*.status.loggedIn = false;

}

},

getters : {

isLoggedIn: *state* => *state*.user,

}

};

**Приложение 28**

**Код компонента «LoginPage.vue»**

<template>

<main *class*="log-acc">

<div *class*="container">

<form *class*="form-box" *name*="form" @*submit*.*prevent*="handleLogin">

<div *class*="input-box">

<i *class*="fa-solid fa-user input-box\_\_icon"></i>

<input

*v-model*="user.username"

*class*="input-field"

*type*="text"

*name*="username"

*placeholder*="Логин"

*required*

/>

</div>

<div *class*="input-box">

<i *class*="fa-solid fa-unlock input-box\_\_icon"></i>

<input

*v-model*="user.password"

*class*="input-field"

*type*="password"

*name*="password"

*placeholder*="Пароль"

*required*

/>

</div>

<button *class*="button-simple">Войти</button>

<div *class*="error-message">

<div v-if="message" *role*="alert">

{{ message.message }}

</div>

</div>

</form>

<div *class*="button-simple button-reg">

<router-link *to*="/register" *class*="button-reg\_\_link">

Зарегистрироваться

</router-link>

</div>

</div>

</main>

</template>

<script>

import User from "../../models/User";

export default {

name: "LoginPage",

data() {

return {

user: **new** User("", ""),

loading: false,

message: "",

};

},

computed: {

loggedIn() {

return *this*.$store.state.auth.status.loggedIn;

},

},

created() {

if (*this*.loggedIn) {

*this*.$router.push("/profile");

}

},

methods: {

handleLogin() {

*this*.loading = true;

if (*this*.user.username && *this*.user.password) {

*this*.$store.dispatch("auth/login", *this*.user).then(

() => {

*this*.$router.push("/profile");

},

(*error*) => {

*this*.loading = false;

*this*.message =

(*error*.response && *error*.response.data) ||

*error*.message ||

*error*.toString();

}

);

}

},

},

};

</script>

<style *lang*="scss" *scoped*>

*.background-flex* {

display: flex;

flex-direction: column;

justify-content: space-between;

}

*.form-box* {

width: 50%;

margin: 0 auto;

display: flex;

flex-direction: column;

gap: 15px;

margin-top: 30px;

}

*.input-field* {

height: $button-height;

width: 100%;

background-color: rgba(255, 255, 255, 0.1);

border-radius: 10px;

padding: 15px 20px 15px 60px;

box-sizing: border-box;

border: none;

color: $color-hover;

font-family: $font-main;

font-size: 24px;

font-style: normal;

font-weight: 400;

}

*.button-simple* {

height: $button-height;

background-color: $color-text;

border-radius: 10px;

color: $color-main;

border: 1px solid transparent;

cursor: pointer;

font-family: $font-main;

font-size: 24px;

font-style: normal;

font-weight: 700;

line-height: normal;

box-sizing: border-box;

transition: all 0.3s;

&*:hover* {

background-color: $color-hover;

border: 1px solid $color-text;

}

}

*.button-reg* {

width: 50%;

margin: 0 auto;

display: flex;

align-items: center;

justify-content: center;

margin-bottom: 15px;

&*\_\_link* {

color: $color-main;

text-decoration: none;

}

}

*.input-box* {

position: relative;

&*\_\_icon* {

position: absolute;

font-size: 24px;

top: calc(($button-height - 24px) / 2);

left: 20px;

}

}

*.error-message* {

margin: 0 auto;

height: $button-height;

margin-bottom: 15px;

display: flex;

align-items: center;

font-family: "Cormorant Garamond";

font-size: 24px;

font-style: normal;

font-weight: 700;

line-height: normal;

}

</style>

**Приложение 29**

**Код компонента «RegisterPage.vue»**

<template>

<div *class*="container">

<form *name*="form" @*submit*.*prevent*="handleRegister">

<div v-if="!successful" *class*="form-box">

<div *class*="input-box">

<i *class*="fa-solid fa-user input-box\_\_icon"></i>

<input

*v-model*="user.username"

*type*="text"

*class*="input-field"

*name*="username"

*placeholder*="Логин"

*required*

/>

</div>

<div *class*="input-box">

<i *class*="fa fa-envelope input-box\_\_icon" *aria-hidden*="true"></i>

<input

*v-model*="user.email"

*class*="input-field"

*required*

*type*="email"

*name*="email"

*placeholder*="Email"

/>

</div>

<div *class*="input-box">

<i *class*="fa-solid fa-unlock input-box\_\_icon"></i>

<input

*v-model*="user.password"

*class*="input-field"

*required*

*type*="password"

*name*="password"

*placeholder*="Пароль"

/>

</div>

<div *class*="input-box">

<i *class*="fa-solid fa-user input-box\_\_icon"></i>

<input

*v-model*="user.name"

*class*="input-field"

*type*="text"

*name*="username"

*placeholder*="Ваше имя"

/>

</div>

<div *class*="input-box">

<i *class*="fa fa-phone input-box\_\_icon" *aria-hidden*="true"></i>

<input

*v-model*="user.phone"

*class*="input-field"

*type*="text"

*name*="phone"

*placeholder*="Телефон"

/>

</div>

<div *class*="input-box">

<i *class*="fa fa-birthday-cake input-box\_\_icon" *aria-hidden*="true"></i>

<input

*v-model*="user.birthdate"

*class*="input-field"

*type*="date"

*name*="birthdate"

*placeholder*="Дата рождения"

/>

</div>

<button *class*="button-simple">Зарегистрироваться</button>

</div>

</form>

<div v-if="message" *class*="reg-message">{{ message }}</div>

</div>

</template>

<script>

import User from "../../models/User";

export default {

name: "RegisterPage",

data() {

return {

user: new User("", "", "", "", "", ""),

submitted: false,

successful: false,

message: "",

};

},

computed: {

loggedIn() {

return *this*.$store.state.auth.status.loggedIn;

},

},

mounted() {

if (*this*.loggedIn) {

*this*.$router.push("/profile");

}

},

methods: {

handleRegister() {

*this*.message = "";

*this*.submitted = true;

*this*.$store.dispatch("auth/register", *this*.user).then(

(*data*) => {

*this*.message = *data*.message;

*this*.successful = true;

},

(*error*) => {

*this*.message =

(*error*.response && *error*.response.data) ||

*error*.message ||

*error*.toString();

*this*.successful = false;

}

);

},

},

};

</script>

<style *lang*="scss" *scoped*>

*.form-box* {

width: 50%;

margin: 0 auto;

display: flex;

flex-direction: column;

gap: 15px;

margin-top: 30px;

}

*.input-field* {

height: $button-height;

width: 100%;

background-color: rgba(255, 255, 255, 0.1);

border-radius: 10px;

padding: 15px 20px 15px 60px;

box-sizing: border-box;

border: none;

color: $color-hover;

font-family: $font-main;

font-size: 24px;

font-style: normal;

font-weight: 400;

}

*.input-box* {

position: relative;

&*\_\_icon* {

position: absolute;

font-size: 24px;

top: calc(($button-height - 24px) / 2);

left: 20px;

}

}

*.button-simple* {

height: $button-height;

background-color: $color-text;

border-radius: 10px;

color: $color-main;

border: 1px solid transparent;

cursor: pointer;

font-family: $font-main;

font-size: 24px;

font-style: normal;

font-weight: 700;

line-height: normal;

box-sizing: border-box;

transition: all 0.3s;

margin-bottom: 15px;

&*:hover* {

background-color: $color-hover;

// color: $color-text;

border: 1px solid $color-text;

}

}

*.reg-message*{

text-align: center;

font-size: 24px;

font-weight: 700;

margin-top: 40px;

}

</style>

**Приложение 30**

**Код компонента «AccountPage.vue»**

<template>

<div *class*="account">

<div *class*="container acc-box">

<h3 *class*="acc-title">

Здравствуйте, {{ currentUser.name || currentUser.username }}!

</h3>

<ul *class*="account\_\_profile">

<li *class*="account\_\_list">Номер телефона: {{ currentUser.phone }}</li>

<li *class*="account\_\_list">Email: {{ currentUser.email }}</li>

</ul>

<div *class*="account\_\_buttons">

<router-link *to*="/userShedule" *class*="account\_\_link button-simple"

>Мои записи</router-link

>

<router-link

v-if="!showAdminBoard && !showMasterBoard"

*to*="/shedule"

*class*="account\_\_link button-simple"

>Записаться</router-link

>

<router-link

v-if="showMasterBoard"

*to*="/master"

*class*="account\_\_link button-simple"

>Записи ко мне</router-link

>

<router-link

v-if="showAdminBoard"

*to*="/admin"

*class*="account\_\_link button-simple"

>Админ</router-link

>

</div>

<a @*click*="logOut" *class*="account\_\_link button-simple" *href*="#">

<span>Выйти</span>

</a>

</div>

</div>

</template>

<script>

export default {

name: "AccountPage",

computed: {

currentUser() {

return *this*.$store.state.auth.user;

},

showMasterBoard() {

if (*this*.currentUser && *this*.currentUser.roles) {

return *this*.currentUser.roles.includes("ROLE\_MASTER");

}

return false;

},

showAdminBoard() {

if (*this*.currentUser && *this*.currentUser.roles) {

return *this*.currentUser.roles.includes("ROLE\_ADMIN");

}

return false;

},

},

mounted() {

if (!*this*.currentUser) {

*this*.$router.push("/login");

}

},

methods: {

logOut() {

*this*.$store.dispatch("auth/logout");

*this*.$router.push("/login");

},

},

};

</script>

<style *lang*="scss" *scoped*>

*.acc-title* {

margin-bottom: 32px;

margin-top: 40px;

text-align: center;

font-size: 32px;

}

*.acc-box* {

width: 50%;

margin: 0 auto;

}

*.account* {

&*\_\_profile* {

list-style-type: none;

font-family: "Cormorant Garamond";

font-size: 24px;

font-style: normal;

}

&*\_\_list* {

margin-bottom: 15px;

}

&*\_\_buttons* {

display: grid;

grid-template-columns: 1fr 1fr;

gap: 15px;

margin-bottom: 15px;

}

&*\_\_link* {

text-decoration: none;

display: flex;

align-items: center;

justify-content: center;

transition: all 0.3s;

&*:hover* {

background-color: $color-hover;

// color: $color-text;

border: 1px solid $color-text;

}

}

*.button-simple* {

height: 60px;

background-color: #cdaa7d;

border-radius: 10px;

color: #0a1111;

border: 1px solid transparent;

cursor: pointer;

font-family: "Cormorant Garamond";

font-size: 24px;

font-style: normal;

font-weight: 700;

line-height: normal;

box-sizing: border-box;

transition: all 0.3s;

}

*.button-simple:hover* {

background-color: #ffffff;

border: 1px solid #cdaa7d;

}

}

</style>

**Приложение 31**

**Код компонента «SheduleUser.vue»**

<template>

<div *class*="container">

<h2 *class*="pages\_\_title">Мои записи</h2>

<div *class*="shedule-lk">

<ul >

<li

v-for="shedule in separationShedule.future"

:*key*="shedule.id"

*class*="shedule-lk\_\_list shedule-lk\_\_row"

>

<div >

{{ printDate(shedule.date) }}

в

{{ shedule.time }}

к мастеру:

{{ shedule.master.name }}

на

{{ shedule.service.title }} - стоимость:

{{ shedule.service.price }} руб.

</div>

<div @*click*="deleteShedule(shedule.id)" *class*="shedule-lk\_\_delete">

<i *class*="fa fa-trash" *aria-hidden*="true"></i>

</div>

</li>

</ul>

<div v-if="separationShedule.future.length === 0">

<router-link *to*="/shedule" *class*="shedule-lk\_\_link button-simple"

>Записаться</router-link

>

</div>

<h3 *class*="shedule-lk\_\_title">Последние записи:</h3>

<ul>

<li

v-for="shedule in separationShedule.past"

:*key*="shedule.id"

*class*="shedule-lk\_\_list shedule-lk\_\_list-past "

>

{{ printDate(shedule.date) }}

в

{{ shedule.time }}

к мастеру:

{{ shedule.master.name }}

на

{{ shedule.service.title }}

</li>

</ul>

</div>

</div>

</template>

<script>

import { mapGetters } from "vuex";

import utils from "../../services/utils";

export default {

name: "SheduleUser",

computed: {

...mapGetters(["SHEDULE"]),

currentUser() {

return *this*.$store.state.auth.user;

},

dateNow() {

return **new** *Date*().getTime();

},

separationShedule() {

const separation = { past: [], future: [] };

const countPastShedules = 5;

*this*.SHEDULE.forEach((*shedule*) => {

const dateShedule = **new** *Date*(

`${*shedule*.date} ${*shedule*.time}`

).getTime();

if (dateShedule > *this*.dateNow) {

separation.future.push(*shedule*);

} else {

separation.past.push(*shedule*);

}

});

separation.future.sort((*a*, *b*) => {

const date1 = **new** *Date*(`${*a*.date} ${*a*.time}`).getTime();

const date2 = **new** *Date*(`${*b*.date} ${*b*.time}`).getTime();

return date1 - date2;

});

separation.past.sort((*a*, *b*) => {

const date1 = **new** *Date*(`${*a*.date} ${*a*.time}`).getTime();

const date2 = **new** *Date*(`${*b*.date} ${*b*.time}`).getTime();

return date2 - date1;

});

separation.past.splice(

countPastShedules - 1,

separation.past.length - countPastShedules

);

return separation;

},

},

mounted() {

*this*.$store.dispatch("GET\_SHEDULE\_BY\_USER", *this*.currentUser.id);

},

methods: {

deleteShedule(*id*) {

const answer = confirm("Удалить запись?");

if (answer) {

*this*.$store.dispatch("DELETE\_SHEDULE", *id*);

alert("Запись удалена");

}

},

printDate(*dateStr*) {

return utils.printDate(*dateStr*);

},

},

};

</script>

<style *lang*="scss" *scoped*>

@import "../../styles/shedule";

*.pages\_\_title* {

font-weight: 700;

font-size: 48px;

line-height: 120%;

text-align: center;

margin-top: 40px;

margin-bottom: 20px;

}

*.button-simple* {

height: 60px;

background-color: #cdaa7d;

border-radius: 10px;

color: #0a1111;

border: 1px solid transparent;

cursor: pointer;

font-family: "Cormorant Garamond";

font-size: 24px;

font-style: normal;

font-weight: 700;

line-height: normal;

box-sizing: border-box;

transition: all 0.3s;

&*:hover* {

background-color: #ffffff;

border: 1px solid #cdaa7d;

}

}

</style>

**Приложение 32**

**Код модуля хранилища «shedule.module.js»**

import Shedule from "../models/Shedule";

import Master from "../models/Master";

import Service from "../models/Service";

import User from "../models/User";

import shedulesService from "../services/shedules.service";

export const sheduleStore = {

  state: {

    shedule: [],

  },

  getters: {

    SHEDULE(*state*) {

      return *state*.shedule;

    },

  },

  mutations: {

    SET\_SHEDULE(*state*, *sheduleList*) {

*state*.shedule = *sheduleList*;

    },

    ADD\_SHEDULE: (*state*, *shedule*) => {

*state*.shedule.push(*shedule*);

    },

    DELETE\_SHEDULE: (*state*, *id*) => {

*state*.shedule.splice(

*state*.shedule.findIndex((*el*) => *el*.id === *id*),

        1

      );

    },

  },

  actions: {

    GET\_SHEDULE: async (*context*) => {

      shedulesService

        .getAllShedule()

        .then((*res*) => {

          const data = [];

*res*.data.forEach((*element*) => {

            data.push(

              new Shedule(

*element*.id,

*element*.date,

*element*.time,

*element*.userId,

*element*.masterId,

*element*.serviceId

              )

            );

          });

*context*.commit("SET\_SHEDULE", data);

        })

        .catch((*e*) => {

          console.log(*e*);

        });

    },

    GET\_SHEDULE\_BY\_DATE\_AND\_MASTER: async (*context*, *param*) => {

      shedulesService

        .getShedulesByDateAndMaster(*param*)

        .then((*res*) => {

          const data = [];

*res*.data.forEach((*element*) => {

            data.push(

              new Shedule(

*element*.id,

*element*.date,

*element*.time,

*element*.userId,

*element*.masterId,

*element*.serviceId

              )

            );

          });

*context*.commit("SET\_SHEDULE", data);

        })

        .catch((*e*) => {

          console.log(*e*);

        });

    },

    GET\_SHEDULE\_BY\_DATE\_AND\_MASTER\_FOR\_MASTER: async (*context*, *params*) => {

      shedulesService

        .getSheduleForMaster(*params*)

        .then((*res*) => {

          const data = [];

*res*.data.forEach((*element*) => {

            const user = new User(

*element*.userId,

              null,

              null,

*element*.user.name

            );

            const service = new Service(

*element*.serviceId,

*element*.service.title

            );

            data.push(

              new Shedule(

*element*.id,

*element*.date,

*element*.time,

*element*.userId,

*element*.masterId,

*element*.serviceId,

                null,

                service,

                user

              )

            );

          });

*context*.commit("SET\_SHEDULE", data);

        })

        .catch((*e*) => {

          console.log(*e*);

        });

    },

    GET\_SHEDULE\_BY\_DATE\_FOR\_ADMIN: async (context, date) => {

      shedulesService

        .getSheduleForAdmin(*date*)

        .then((*res*) => {

          const data = [];

*res*.data.forEach((*element*) => {

            const user = new User(

*element*.userId,

              null,

              null,

              null,

*element*.user.name,

*element*.user.phone

            );

            const master = new Master(

*element*.masterId,

*element*.master.master\_name,

              null,

              null,

*element*.master.position

            );

            const service = new Service(

*element*.serviceId,

*element*.service.title,

              null,

*element*.service.duration,

*element*.service.price

            );

            data.push(

              new Shedule(

*element*.id,

*element*.date,

*element*.time,

*element*.userId,

*element*.masterId,

*element*.serviceId,

                master,

                service,

                user

              )

            );

          });

*context*.commit("SET\_SHEDULE", data);

        })

        .catch((*e*) => {

          console.log(*e*);

        });

    },

    CREATE\_SHEDULE: async (*context*, *data*) => {

      shedulesService

        .createShedule(*data*)

        .then((*res*) => {

          console.log(*res*.data);

        })

        .catch((*e*) => {

          console.log(e);

        });

    },

    GET\_SHEDULE\_BY\_USER: async (context, userId) => {

      shedulesService

        .getSheduleByUser(userId)

        .then((res) => {

          const data = [];

          res.data.forEach((element) => {

            const master = new *Master*(

              element.masterId,

              element.master.master\_name

            );

            const service = new *Service*(

              element.serviceId,

              element.service.title,

              null,

              element.service.duration,

              element.service.price

            );

            data.push(

              new *Shedule*(

                element.id,

                element.date,

                element.time,

                element.userId,

                element.masterId,

                element.serviceId,

                master,

                service

              )

            );

          });

          context.commit("SET\_SHEDULE", data);

        })

        .catch((e) => {

*console*.log(e);

        });

    },

    DELETE\_SHEDULE: async (context, id) => {

      shedulesService

        .deleteSheduleById(id)

        .then((res) => {

*console*.log(res.data);

          context.commit("DELETE\_SHEDULE", id);

        })

        .catch((e) => {

*console*.log(e);

        });

    },

  },

};

**Приложение 33**

**Код сервиса «shedules.service.js»**

import axios from "axios";

const API\_URL = "http://localhost:8080/api/shedule";

class SheduleService {

  getAllShedule() {

    return axios.get(API\_URL);

  }

  getShedulesByDateAndMaster(*param*) {

    return axios.get(API\_URL + `?date=${*param*.date}&masterId=${*param*.id}`);

  }

  createShedule(*data*) {

    return axios.post(API\_URL, *data*);

  }

  getSheduleByUser(*userId*) {

    return axios.get(API\_URL + `/userShedule/list?userId=${*userId*}`);

  }

  getSheduleForMaster(*params*) {

    return axios.get(

      API\_URL +

        `/masterShedule/list?date=${*params*.date}&masterId=${*params*.masterId}`

    );

  }

  getSheduleForAdmin(*date*) {

    return axios.get(API\_URL + `/adminShedule/list?date=${*date*}`);

  }

  deleteSheduleById(*sheduleId*) {

    return axios.delete(API\_URL + `/userShedule/list/${*sheduleId*}`);

  }

}

export default new SheduleService();

**Приложение 34**

**Код компонента «MasterBoard.vue»**

<template>

  <div *class*="container">

    <div *class*="shedule">

      <div v-if="MASTER\_ACCOUNT" *class*="shedule\_\_title">

        <label *for*="dateMasterShedule">Выберите дату: </label>

        <input *type*="date" *id*="dateMasterShedule" *v-model*.*lazy*="selectedDate" />

      </div>

      <div v-if="SHEDULE?.length !== 0">

        <p v-for="item in SHEDULE" :*key*="item.id">

          {{ item.time }} {{ item.user.name }} на {{ item.service.title }}

        </p>

      </div>

      <div v-if="!MASTER\_ACCOUNT">

        Мастера с вашим именем и фамилией не найдено

      </div>

      <div v-if="SHEDULE?.length === 0">

        Записи на выбранную дату отсутствуют

      </div>

    </div>

  </div>

</template>

<script>

import { mapGetters } from "vuex";

import utils from "../../services/utils";

export default {

  name: "MasterBoard",

  data() {

    return {

      selectedDate: utils.getDate(**new** *Date*()),

    };

  },

  mounted() {

    const currentUser = *this*.$store.state.auth.user;

    const param = { name: currentUser.name, phone: currentUser.phone };

*this*.$store.dispatch("GET\_MASTER\_BY\_NAME\_AND\_PHONE", param).then((*res*) => {

*this*.$store.dispatch("GET\_SHEDULE\_BY\_DATE\_AND\_MASTER\_FOR\_MASTER", {

        date: utils.getDate(new *Date*()),

        masterId: *res*.id,

      });

    });

  },

  watch: {

    selectedDate: function () {

      const param = {

        date: *this*.selectedDate,

        masterId: *this*.MASTER\_ACCOUNT.id,

      };

*this*.$store.dispatch("GET\_SHEDULE\_BY\_DATE\_AND\_MASTER\_FOR\_MASTER", param);

    },

  },

  computed: {

    ...mapGetters(["SHEDULE", "MASTER\_ACCOUNT"]),

    currentUser() {

      return *this*.$store.state.auth.user;

    },

  },

};

</script>

<style *lang*="scss" *scoped*>

@import "../../styles/shedule";

*.pages\_\_title* {

  font-weight: 700;

  font-size: 48px;

  line-height: 120%;

  text-align: center;

  margin-top: 40px;

  margin-bottom: 20px;

}

</style>

**Приложение 35**

**Код компонента «AdminBoard.vue»**

<template>

  <div *class*="container">

    <div *class*="adminBoard">

      <router-link *to*="/admin/usersList" *class*="adminBoard\_\_link button-simple"

        >Клиенты</router-link

      >

      <router-link

*to*="/admin/mastersList"

*class*="adminBoard\_\_link button-simple"

        >Мастера

      </router-link>

      <router-link

*to*="/admin/servicesList"

*class*="adminBoard\_\_link button-simple"

        >Услуги</router-link

      >

      <router-link

*to*="/admin/shedulesList"

*class*="adminBoard\_\_link button-simple"

        >Записи</router-link

      >

    </div>

  </div>

</template>

<script>

export default {

  name: "AdminBoard",

};

</script>

<style *lang*="scss" *scoped*>

*.button-simple* {

  height: 60px;

  background-color: #cdaa7d;

  border-radius: 10px;

  color: #0a1111;

  border: 1px solid transparent;

  cursor: pointer;

  font-family: "Cormorant Garamond";

  font-size: 24px;

  font-style: normal;

  font-weight: 700;

  line-height: normal;

  box-sizing: border-box;

  transition: all 0.3s;

  &*:hover* {

    background-color: $color-hover;

    border: 1px solid $color-text;

  }

}

*.adminBoard* {

  padding: 20px;

  width: 50%;

  margin: 0 auto;

  &*\_\_link* {

    text-decoration: none;

    display: flex;

    align-items: center;

    justify-content: center;

    margin-bottom: 20px;

    transition: all 0.3s;

  }

}

</style>

**Приложение 36**

**Код компонента «AdminUsers.vue»**

<template>

  <div *class*="container">

    <div *class*="shedule">

      <div *class*="shedule\_\_title">Наши клиенты</div>

      <div>

        <label *for*="userPhone" *class*="shedule\_\_title"

          >Поиск по номеру телефона:

        </label>

        <input *type*="text" *id*="userPhone" *v-model*.*lazy*="inputPhone" />

        <i  *class*="fa fa-search button-search" *aria-hidden*="true"></i>

      </div>

      <table *class*="shedule-table">

        <thead>

          <tr *class*="shedule-table\_\_row">

            <th>Имя</th>

            <th>Email</th>

            <th>Телефон</th>

            <th>Дата рождения</th>

            <th>Действия</th>

          </tr>

        </thead>

        <tbody>

          <tr *class*="shedule-table\_\_row" v-for="user in USERS" :*key*="user.id">

            <td>{{ user.name }}</td>

            <td>{{ user.email }}</td>

            <td>{{ user.phone }}</td>

            <td>{{ user.birthdate }}</td>

            <td>

              <i @*click*="makeAppointmentUser(user.id)"

*class*="fa fa-pencil-square-o shedule-lk\_\_delete"

*aria-hidden*="true"

              ></i>

              <i

                @*click*="deleteUser(user.id)"

*class*="fa fa-trash shedule-lk\_\_delete"

*aria-hidden*="true"

              ></i>

            </td>

          </tr>

        </tbody>

      </table>

      <div v-if="USERS?.length === 0">Клиенты отсутсвуют</div>

      <div *class*="arrows">

        <i

          v-if="start <= 0"

*class*="fa fa-circle arrows\_\_item"

*aria-hidden*="true"

        ></i>

        <i

          @*click*="previousList()"

*class*="fa fa-chevron-circle-left arrows\_\_item"

*aria-hidden*="true"

          v-if="start > 0"

        ></i>

        <i

          @*click*="nextList()"

*class*="fa fa-chevron-circle-right arrows\_\_item"

*aria-hidden*="true"

          v-if="remainder > 0"

        ></i>

        <i

*class*="fa fa-circle arrows\_\_item"

*aria-hidden*="true"

          v-if="remainder <= 0"

        ></i>

      </div>

    </div>

  </div>

</template>

<script>

import { mapGetters, mapMutations } from "vuex";

export default {

  name: "AdminUsers",

  data() {

    return {

      start: 0,

      count: 5,

      inputPhone: null,

    };

  },

  mounted() {

    const params = { start: 0, count: *this*.count };

*this*.$store.dispatch("GET\_USERS", params);

*this*.$store.dispatch("GET\_USER\_COUNT");

  },

  watch: {

    inputPhone: function () {

*this*.start = 0;

      const params = {

        start: *this*.start,

        count: *this*.count,

        phone: *this*.inputPhone,

      };

      console.log(params);

*this*.$store.dispatch("GET\_USERS", params);

*this*.$store.dispatch("GET\_USER\_COUNT", *this*.inputPhone);

    },

  },

  computed: {

    ...mapGetters(["USERS", "USERS\_COUNT"]),

    remainder() {

      return *this*.USERS\_COUNT - (*this*.start / *this*.count + 1) \* *this*.count;

    },

  },

  methods: {

    ...mapMutations(["SET\_USER\_FOR\_SHEDULE"]),

    deleteUser(*id*) {

      console.log("del");

      const answer = confirm("Удалить аккаунт клиента?");

      if (answer) {

*this*.$store.dispatch("DELETE\_USER", *id*);

        alert("Запись удалена");

      }

    },

    nextList() {

      let params;

      if (*this*.inputPhone) {

        params = {

          start: *this*.start + *this*.count,

          count: *this*.count,

          phone: *this*.inputPhone,

        };

      } else {

        params = { start: *this*.start + *this*.count, count: *this*.count };

      }

*this*.$store.dispatch("GET\_USERS", params);

*this*.start += *this*.count;

    },

    previousList() {

      let params;

      if (*this*.inputPhone) {

        params = {

          start: *this*.start - *this*.count,

          count: *this*.count,

          phone: *this*.inputPhone,

        };

      } else {

        params = { start: *this*.start - *this*.count, count: *this*.count };

      }

*this*.$store.dispatch("GET\_USERS", params);

*this*.start -= *this*.count;

    },

    makeAppointmentUser(*id*){

*this*.SET\_USER\_FOR\_SHEDULE(*id*);

*this*.$router.push({ path: `/shedule` });

    }

  },

};

</script>

<style *lang*="scss" *scoped*>

@import "../../styles/shedule";

*.arrows* {

  width: 70px;

  display: flex;

  justify-content: space-between;

  &*\_\_item* {

    cursor: pointer;

    &*:hover* {

      color: $color-hover;

    }

  }

}

*.button-search*{

  cursor: pointer;

  margin-left: 15px;

  &*:hover* {

      color: $color-hover;

    }

}

</style>

**Приложение 37**

**Код модуля хранилища «user.module.js»**

import User from '../models/User';

import userService from "../services/user.service"

export const userStore = {

    state: {

        users: [],

        userCount: null,

        userForShedule: null,

    },

    getters: {

        USERS(*state*) {

            return *state*.users;

        },

        USERS\_COUNT(*state*) {

            return *state*.userCount;

        },

        USER\_FOR\_SHEDULE(*state*){

            return *state*.userForShedule;

        }

    },

    mutations: {

        SET\_USERS(*state*, *usersList*) {

*state*.users = *usersList*;

        },

        ADD\_USER: (*state*, *user*) => {

*state*.users.push(*user*);

        },

        DELETE\_USER: (*state*, *id*) => {

*state*.users.splice(*state*.users.findIndex(*el* => *el*.id === *id*), 1);

        },

        SET\_USERS\_COUNT(*state*, *count*) {

*state*.userCount = *count*;

        },

        SET\_USER\_FOR\_SHEDULE(*state*, *id*) {

*state*.userForShedule = *id*;

        },

    },

    actions: {

        GET\_USERS: async (*context*, *params*) => {

            userService.getAllUsersForAdmin(*params*).then((*res*) => {

                const data = [];

*res*.data.forEach((*user*) => {

                    data.push(**new** User(*user*.id, null, *user*.email, null, *user*.name, *user*.phone, *user*.birthdate));

                });

                console.log(data);

*context*.commit('SET\_USERS', data);

            })

                .catch((*e*) => {

                    console.log(*e*);

                });

        },

        GET\_USER\_COUNT: async (*context*, *phone*) => {

            userService.getUsersCount(*phone*).then((*res*) => {

*context*.commit('SET\_USERS\_COUNT', *res*.data);

            })

                .catch((*e*) => {

                    console.log(*e*);

                });

        },

        DELETE\_USER: async (*context*, *id*) => {

            userService.deleteUser(*id*).then((*res*) => {

                console.log(*res*.data);

*context*.commit('DELETE\_USER', *id*);

            })

                .catch((*e*) => {

                    console.log(*e*);

                });

        },

    },

}

**Приложение 38**

**Код компонента «AdminMasters.vue»**

<template>

  <div *class*="container">

    <div *class*="shedule">

      <div *class*="shedule\_\_title">Мастера</div>

      <table *class*="shedule-table">

        <thead>

          <tr *class*="shedule-table\_\_row">

            <th>Имя</th>

            <th>Должность</th>

            <th>Телефон</th>

            <th>Дата рождения</th>

            <th>Фото</th>

            <th>О мастере</th>

            <th>Действия</th>

          </tr>

        </thead>

        <tbody>

          <tr *class*="shedule-table\_\_row" v-for="item in MASTERS" :*key*="item.id">

            <td>

              <div v-if="!needEdit || editId != item.id">{{ item.name }}</div>

              <div v-if="needEdit && editId === item.id">

                <input

*type*="text"

*id*="newMasterName"

*v-model*.*lazy*="master.master\_name"

                />

              </div>

            </td>

            <td>

              <div v-if="!needEdit || editId != item.id">{{ item.position }}</div>

              <div v-if="needEdit && editId === item.id">

                <input

*type*="text"

*id*="masterPosition"

*v-model*.*lazy*="master.position"

                />

                <select

*v-model*="master.group\_service\_id"

*name*="group"

*id*="group-select"

                >

                  <option *value*="Выберите группу услуг">

                    -- Выберите группу услуг --

                  </option>

                  <option

*v-bind*:*value*="item.id"

                    v-for="item in GROUP\_SERVICES"

                    :*key*="item.id"

                  >

                    {{ item.title }}

                  </option>

                </select>

              </div>

            </td>

            <td>

              <div v-if="!needEdit || editId != item.id">{{ item.phone }}</div>

              <div v-if="needEdit && editId === item.id">

                <input

*type*="text"

*id*="masterPhone"

*v-model*.*lazy*="master.phone"

                />

              </div>

            </td>

            <td>

              <div v-if="!needEdit || editId != item.id">{{ item.birthdate }}</div>

              <div v-if="needEdit && editId === item.id">

                <input

*type*="date"

*id*="masterBirth"

*v-model*.*lazy*="master.birthdate"

                />

              </div>

            </td>

            <td>

              <div v-if="!needEdit || editId != item.id">{{ item.photo }}</div>

              <div v-if="needEdit && editId === item.id">

                <input

*type*="text"

*id*="masterPhoto"

*v-model*.*lazy*="master.photo"

                />

              </div>

            </td>

            <td>

              <div v-if="!needEdit || editId != item.id">{{ item.aboutText }}</div>

              <div v-if="needEdit && editId === item.id">

                <textarea

*v-model*.*lazy*="master.about\_text"

*id*="masterAbout"

*rows*="3"

                >

                     Инфо о мастере</textarea

                >

              </div>

            </td>

            <td>

              <div v-if="!needEdit || editId != item.id">

                <i

                  @*click*="editMaster(item.id)"

*class*="fa fa-pencil-square-o shedule-lk\_\_delete"

*aria-hidden*="true"

                ></i>

                <i

                  @*click*="deleteMaster(item.id)"

*class*="fa fa-trash shedule-lk\_\_delete"

*aria-hidden*="true"

                ></i>

              </div>

              <div v-if="needEdit && editId === item.id">

                <i

                  @*click*="updateMaster()"

*class*="fa fa-check-square-o"

*aria-hidden*="true"

                ></i>

              </div>

            </td>

          </tr>

          <tr v-if="isShowInput" *class*="shedule-table\_\_row">

            <td>

              <input

*type*="text"

*id*="masterName"

*v-model*.*lazy*="master.master\_name"

              />

            </td>

            <td>

              <input

*type*="text"

*id*="masterPosition"

*v-model*.*lazy*="master.position"

              />

            </td>

            <td>

              <input *type*="text" *id*="masterPhone" *v-model*.*lazy*="master.phone" />

            </td>

            <td>

              <input

*type*="date"

*id*="masterBirth"

*v-model*.*lazy*="master.birthdate"

              />

            </td>

            <td>

              <input *type*="text" *id*="masterPhoto" *v-model*.*lazy*="master.photo" />

            </td>

            <td>

              <textarea

*v-model*.*lazy*="master.about\_text"

*id*="masterAbout"

*rows*="3"

              >

                     Инфо о мастере</textarea

              >

            </td>

            <td>

              <select

*v-model*="master.group\_service\_id"

*name*="group"

*id*="group-select"

              >

                <option *value*="Выберите группу услуг">

                  -- Выберите группу услуг --

                </option>

                <option

*v-bind*:*value*="item.id"

                  v-for="item in GROUP\_SERVICES"

                  :*key*="item.id"

                >

                  {{ item.title }}

                </option>

              </select>

            </td>

          </tr>

        </tbody>

      </table>

      <div v-if="MASTERS?.length === 0">Мастера отсутствуют</div>

      <button

        v-if="!isShowInput && !needEdit"

        @*click*="isShowInput = true"

*class*="subscribeBtn button-margin"

      >

        Добавить нового мастера

      </button>

      <p>{{ message }}</p>

      <button

        v-if="isShowInput"

        @*click*="createMaster()"

*class*="subscribeBtn button-margin"

      >

        Сохранить мастера

      </button>

    </div>

  </div>

</template>

<script>

import { mapGetters } from "vuex";

export default {

  name: "AdminMasters",

  data() {

    return {

      isShowInput: false,

      master: {

        master\_name: null,

        position: null,

        phone: null,

        birthdate: null,

        photo: null,

        about\_text: null,

        group\_service\_id: null,

      },

      message: null,

      editId: null,

      needEdit: false,

    };

  },

  mounted() {

*this*.$store.dispatch("GET\_MASTERS");

  },

  computed: {

    ...mapGetters(["MASTERS", "GROUP\_SERVICES", "MASTER\_BY\_ID"]),

  },

  methods: {

    deleteMaster(id) {

      const answer = confirm("Удалить аккаунт мастера?");

      if (answer) {

*this*.$store.dispatch("DELETE\_MASTER", id);

        alert("Запись удалена");

      }

    },

    editMaster(id) {

*console*.log("edit", id);

*this*.editId = id;

*this*.needEdit = true;

      const masterForEdit = *this*.MASTER\_BY\_ID(id);

*this*.master.master\_name = masterForEdit.name;

*this*.master.position = masterForEdit.position;

*this*.master.phone = masterForEdit.phone;

*this*.master.birthdate = masterForEdit.birthdate;

*this*.master.photo = masterForEdit.photo;

*this*.master.about\_text = masterForEdit.aboutText;

*this*.master.group\_service\_id = masterForEdit.groupServiceId;

    },

    updateMaster() {

      if (*Object*.values(*this*.master).every((value) => !!value)) {

*this*.$store

          .dispatch("UPDATE\_MASTER", { id: *this*.editId, master: *this*.master })

          .then(() => *this*.$store.dispatch("GET\_MASTERS"));

*this*.needEdit = false;

*this*.message = null;

      } else {

*this*.message = "Заполнены не все поля!";

      }

    },

    createMaster() {

      if (*Object*.values(*this*.master).every((value) => !!value)) {

*this*.$store.dispatch("CREATE\_MASTER", *this*.master);

*this*.isShowInput = false;

*this*.message = null;

      } else {

*this*.message = "Заполнены не все поля!";

      }

    },

  },

};

</script>

<style *lang*="scss" *scoped*>

@import "../../styles/shedule";

*.subscribeBtn* {

  height: 60px;

  background-color: #cdaa7d;

  border-radius: 10px;

  color: #0a1111;

  border: 1px solid transparent;

  cursor: pointer;

  font-family: "Cormorant Garamond";

  font-size: 24px;

  font-style: normal;

  font-weight: 700;

  line-height: 60px;

  box-sizing: border-box;

  text-decoration: none;

  padding-left: 30px;

  padding-right: 30px;

  transition: all 0.3s;

  &*:hover* {

    background-color: white;

    border: 1px solid #cdaa7d;

  }

}

*.button-margin* {

  margin-top: 30px;

}

</style>

**Приложение 39**

**Код компонента «AdminServices.vue»**

<template>

  <div *class*="container">

    <div *class*="shedule">

      <div *class*="shedule\_\_title">Услуги</div>

      <table *class*="shedule-table">

        <thead>

          <tr *class*="shedule-table\_\_row">

            <th>Услуга</th>

            <th>Длительность</th>

            <th>Цена</th>

            <th>Группа</th>

            <th>Действия</th>

          </tr>

        </thead>

        <tbody>

          <tr

*class*="shedule-table\_\_row"

            v-for="item in SERVICES"

            :*key*="item.id"

          >

            <td>

              <div v-if="!needEdit || editId != item.id">{{ item.title }}</div>

              <div v-if="needEdit && editId === item.id">

                <input *type*="text" *id*="newTitle" *v-model*.*lazy*="service.title" />

              </div>

            </td>

            <td>

              <div>

                {{ item.duration }} мин

              </div>

            </td>

            <td>

              <div v-if="!needEdit || editId != item.id">{{ item.price }}</div>

              <div v-if="needEdit && editId === item.id">

                <input

*type*="text"

*id*="servicePrice"

*v-model*.*lazy*="service.price"

                />

              </div>

            </td>

            <td>

              <div v-if="!needEdit || editId != item.id">

                {{ GET\_GROUP\_SERVICE\_FOR\_ID(item.groupServiceId).title }}

              </div>

              <div v-if="needEdit && editId === item.id">

                <select

*v-model*="service.group\_service\_id"

*name*="serviceGroup"

*id*="serviceGroup"

                >

                  <option *value*="">-- Выберите группу услуг --</option>

                  <option

                    v-for="groupServ in GROUP\_SERVICES"

                    :*key*="groupServ.id"

*v-bind*:*value*="groupServ.id"

                  >

                    {{ groupServ.title }}

                  </option>

                </select>

              </div>

            </td>

            <td>

              <div v-if="!needEdit || editId != item.id">

                <i

                  @*click*="editService(item.id)"

*class*="fa fa-pencil-square-o shedule-lk\_\_actions"

*aria-hidden*="true"

                ></i>

                <i

                  @*click*="deleteService(item.id)"

*class*="fa fa-trash shedule-lk\_\_actions shedule-lk\_\_actions-del"

*aria-hidden*="true"

                ></i>

              </div>

              <div v-if="needEdit && editId === item.id">

                <i

                  @*click*="updateService()"

*class*="fa fa-check-square-o"

*aria-hidden*="true"

                ></i>

              </div>

            </td>

          </tr>

          <tr v-if="isShowInput" *class*="shedule-table\_\_row">

            <td>

              <input

*type*="text"

*id*="serviceName"

*v-model*.*lazy*="service.title"

              />

            </td>

            <td>

              <input

*type*="text"

*id*="serviceDuration"

*v-model*.*lazy*="service.duration"

              />

            </td>

            <td>

              <input

*type*="text"

*id*="servicePrice"

*v-model*.*lazy*="service.price"

              />

            </td>

            <td>

              <select

*v-model*="service.group\_service\_id"

*name*="group"

*id*="group-select-serv"

              >

                <option *value*="Выберите группу услуг">

                  -- Выберите группу услуг --

                </option>

                <option

*v-bind*:*value*="groupService.id"

                  v-for="groupService in GROUP\_SERVICES"

                  :*key*="groupService.id"

                >

                  {{ groupService.title }}

                </option>

              </select>

            </td>

          </tr>

        </tbody>

      </table>

      <div v-if="SERVICES?.length === 0">Услуги отсутствуют</div>

      <button

        v-if="!isShowInput && !needEdit"

        @*click*="isShowInput = true"

*class*="subscribeBtn button-margin"

      >

        Добавить новую услугу

      </button>

      <p>{{ message }}</p>

      <button

        v-if="isShowInput"

        @*click*="createService()"

*class*="subscribeBtn button-margin"

      >

        Сохранить услугу

      </button>

    </div>

  </div>

</template>

<script>

import { mapGetters } from "vuex";

export default {

  name: "AdminServices",

  data() {

    return {

      isShowInput: false,

      service: {

        title: null,

        duration: null,

        price: null,

        group\_service\_id: null,

      },

      message: null,

      editId: null,

      needEdit: false,

    };

  },

  mounted() {

*this*.$store.dispatch("GET\_SERVICES");

  },

  computed: {

    ...mapGetters(["SERVICES", "GROUP\_SERVICES", "GET\_GROUP\_SERVICE\_FOR\_ID", "SERVICE\_FOR\_ID"]),

  },

  methods: {

    deleteService(*id*) {

      const answer = confirm("Удалить услугу?");

      if (answer) {

*this*.$store.dispatch("DELETE\_SERVICE", *id*);

        alert("Запись удалена");

      }

    },

    editService(*id*) {

*this*.editId = *id*;

*this*.needEdit = true;

      const serviceForEdit = *this*.SERVICE\_FOR\_ID(*id*);

*this*.service.title = serviceForEdit.title;

*this*.service.duration = serviceForEdit.duration;

*this*.service.price = serviceForEdit.price;

*this*.service.group\_service\_id = serviceForEdit.groupServiceId;

    },

    updateService() {

      if (*Object*.values(*this*.service).every((*value*) => !!*value*)) {

*this*.$store

          .dispatch("UPDATE\_SERVICE", {

            id: *this*.editId,

            service: *this*.service,

          })

          .then(() => *this*.$store.dispatch("GET\_SERVICES"));

*this*.needEdit = false;

*this*.message = null;

      } else {

*this*.message = "Заполнены не все поля!";

      }

    },

    createService() {

      if (*Object*.values(*this*.service).every((*value*) => !!*value*)) {

*this*.$store.dispatch("CREATE\_SERVICE", *this*.service);

*this*.isShowInput = false;

*this*.message = null;

      } else {

*this*.message = "Заполнены не все поля!";

      }

    },

  },

};

</script>

<style *lang*="scss" *scoped*>

@import "../../styles/shedule";

*.subscribeBtn* {

  height: 60px;

  background-color: #cdaa7d;

  border-radius: 10px;

  color: #0a1111;

  border: 1px solid transparent;

  cursor: pointer;

  font-family: "Cormorant Garamond";

  font-size: 24px;

  font-style: normal;

  font-weight: 700;

  line-height: 60px;

  box-sizing: border-box;

  text-decoration: none;

  padding-left: 30px;

  padding-right: 30px;

  transition: all 0.3s;

  &*:hover* {

    background-color: white;

    border: 1px solid #cdaa7d;

  }

}

*.button-margin* {

  margin-top: 30px;

}

</style>

**Приложение 40**

**Код компонента «AdminShedules.vue»**

<template>

  <div *class*="container">

    <div *class*="shedule">

      <div *class*="shedule\_\_title">

        <label *for*="dateAdminShedule">Выберите дату: </label>

        <input *type*="date" *id*="dateAdminShedule" *v-model*.*lazy*="selectedDate" />

      </div>

      <table *class*="shedule-table">

        <thead>

          <tr *class*="shedule-table\_\_row">

            <th *rowspan*="2">Время</th>

            <th *colspan*="2">Мастер</th>

            <th *colspan*="3">Услуга</th>

            <th *colspan*="2">Клиент</th>

            <th *rowspan*="2">Действие</th>

          </tr>

          <tr *class*="shedule-table\_\_row">

            <td>Имя</td>

            <td>Должность</td>

            <td>Услуга</td>

            <td>Длительность</td>

            <td>Цена</td>

            <td>Имя</td>

            <td>Телефон</td>

          </tr>

        </thead>

        <tbody>

          <tr

*class*="shedule-table\_\_row"

            v-for="shedule in SHEDULE"

            :*key*="shedule.id"

          >

            <td>{{ shedule.time }}</td>

            <td>{{ shedule.master.name }}</td>

            <td>{{ shedule.master.position }}</td>

            <td>{{ shedule.service.title }}</td>

            <td>{{ shedule.service.duration }}мин</td>

            <td>{{ shedule.service.price }} руб.</td>

            <td>{{ shedule.user.name }}</td>

            <td>{{ shedule.user.phone }}</td>

            <td @*click*="deleteShedule(shedule.id)" *class*="shedule-lk\_\_delete">

              <i *class*="fa fa-trash" *aria-hidden*="true"></i>

            </td>

          </tr>

        </tbody>

      </table>

      <div v-if="SHEDULE?.length === 0">

        Записи на выбранную дату отсутствуют

      </div>

    </div>

  </div>

</template>

<script>

import { mapGetters } from "vuex";

import utils from "../../services/utils";

export default {

  name: "AdminShedules",

  data() {

    return {

      selectedDate: utils.getDate(**new** *Date*()),

    };

  },

  mounted() {

*this*.$store.dispatch(

      "GET\_SHEDULE\_BY\_DATE\_FOR\_ADMIN",

      utils.getDate(**new** *Date*())

    );

  },

  watch: {

    selectedDate: function () {

*this*.$store.dispatch("GET\_SHEDULE\_BY\_DATE\_FOR\_ADMIN", *this*.selectedDate);

    },

  },

  computed: {

    ...mapGetters(["SHEDULE"]),

  },

  methods: {

    deleteShedule(*id*) {

      const answer = confirm("Удалить запись?");

      if (answer) {

*this*.$store.dispatch("DELETE\_SHEDULE", *id*);

        alert("Запись удалена");

      }

    },

  },

};

</script>

<style *lang*="scss" *scoped*>

@import "../../styles/shedule";

</style>

**Приложение 41**

**Код компонента «AppointmentPage.vue»**

<template>

  <div class="container shedule">

    <h3>Записаться:</h3>

    <router-link *to*="/shedule/master" *class*="button-simple button-link"

      >Выбрать мастера</router-link

    >

    <router-link *to*="/shedule/service" *class*="button-simple button-link"

      >Выбрать услугу</router-link

    >

  </div>

</template>

<script>

export default {

  name: "AppointmentPage",

};

</script>

<style *lang*="scss" *scoped*>

@import "../../styles/shedule";

*.button-link* {

  text-decoration: none;

  display: flex;

  align-items: center;

  justify-content: center;

  width: 30%;

}

</style>

**Приложение 42**

**Код компонента «AppointmentMaster.vue»**

<<template>

  <div *class*="container shedule">

    <h3>Выберите мастера:</h3>

    <button

      v-for="master in MASTERS"

      :*key*="master.id"

*class*="button-simple button-link"

      @*click*="selectMaster(master)"

    >

      {{ master.name }}

    </button>

  </div>

</template>

<script>

import { mapGetters, mapMutations } from "vuex";

export default {

  name: "AppointmentMaster",

  mounted() {

*this*.$store.dispatch("GET\_MASTERS");

  },

  computed: {

    ...mapGetters(["MASTERS"]),

  },

  methods: {

    ...mapMutations(["SET\_CURRENT\_MASTER"]),

    selectMaster(*master*) {

*this*.SET\_CURRENT\_MASTER(*master*);

*this*.$router.push({ path: `/shedule/master/${*master*.id}` });

    },

  },

};

</script>

<style *lang*="scss" *scoped*>

@import "../../styles/shedule";

</style>

**Приложение 43**

**Код компонента «AppointmentMasterServ.vue»**

<template>

  <div *class*="container shedule">

    <h3>Выберите услугу:</h3>

    <button

      v-for="service in SERVICES"

      :*key*="service.id"

*class*="button-simple button-link"

      @*click*="selectService(service)"

    >

      {{ service.title }}

    </button>

  </div>

</template>

<script>

import { mapGetters, mapMutations } from "vuex";

export default {

  name: "AppointmentMasterServ",

  mounted() {

*this*.$store.dispatch(

      "GET\_SERVICES\_BY\_GROUP",

*this*.CURRENT\_MASTER.groupServiceId

    );

  },

  computed: {

    ...mapGetters(["SERVICES", "CURRENT\_MASTER"]),

  },

  methods: {

    ...mapMutations(["SET\_CURRENT\_SERVICE"]),

    selectService(*service*) {

*this*.SET\_CURRENT\_SERVICE(*service*);

*this*.$router.push({

        path: `/shedule/master/${*this*.CURRENT\_MASTER.id}/${*service*.id}`,

      });

    },

  },

};

</script>

<style *lang*="scss" *scoped*>

@import "../../styles/shedule";

</style>

**Приложение 44**

**Код компонента «AppointmentService.vue»**

<template>

  <div *class*="container shedule">

    <h3>Выберите услугу:</h3>

    <details v-for="groupService in GROUP\_SERVICES" :*key*="groupService.id">

      <summary *class*="summary">{{ groupService.title }}</summary>

      <div *class*="buttons-time-box details">

        <button

          v-for="service in SERVICES.filter(

            (*el*) => *el*.groupServiceId === groupService.id

          )"

          :*key*="service.id"

          @*click*="selectService(service)"

*class*="button-simple button-shedule button-link-services"

        >

          {{ service.title }}

        </button>

      </div>

    </details>

  </div>

</template>

<script>

import { mapGetters, mapMutations } from "vuex";

export default {

  name: "AppointmenService",

  mounted() {

*this*.$store.dispatch("GET\_SERVICES");

  },

  computed: {

    ...mapGetters(["SERVICES", "GROUP\_SERVICES"]),

  },

  methods: {

    ...mapMutations(["SET\_CURRENT\_SERVICE", "SET\_CURRENT\_MASTER"]),

    selectService(*service*) {

*this*.SET\_CURRENT\_SERVICE(*service*);

*this*.SET\_CURRENT\_MASTER(null);

*this*.$router.push({ path: `/shedule/service/${*service*.id}` });

    },

  },

};

</script>

<style *lang*="scss" *scoped*>

@import "../../styles/shedule";

*.summary* {

  text-align: center;

}

*.details* {

  margin-top: 15px;

}

</style>

**Приложение 45**

**Код компонента «MakeAppointment.vue»**

<template>

  <div *class*="container shedule">

    <div>

      <span *class*="shedule\_\_title">Мастер: </span

      ><span *class*="shedule\_\_value">{{

        CURRENT\_MASTER ? CURRENT\_MASTER.name : "Любой мастер"

      }}</span>

    </div>

    <div>

      <span *class*="shedule\_\_title">Услуга: </span

      ><span *class*="shedule\_\_value">{{ CURRENT\_SERVICE.title }}</span>

    </div>

    <div>

      <span *class*="shedule\_\_title">Длительность: </span

      ><span *class*="shedule\_\_value">{{ CURRENT\_SERVICE.duration }} мин</span>

    </div>

    <div>

      <span *class*="shedule\_\_title">Цена: </span

      ><span *class*="shedule\_\_value">{{ CURRENT\_SERVICE.price }} руб.</span>

    </div>

    <div v-if="showSelectMaster" *class*="shedule\_\_select">

      <div *class*="shedule\_\_title">Выберите мастера:</div>

      <button

        v-for="master in MASTERS"

        :*key*="master.id"

        @*click*="setCurrentMaster(master.id)"

*class*="button-simple button-shedule"

        :*class*="CURRENT\_MASTER?.id === master.id ? 'active' : ''"

      >

        {{ master.name }}

      </button>

    </div>

    <div v-if="MASTERS.length === 0" *class*="shedule\_\_box">

      <p *class*="shedule\_\_message">

        Извините, запись на выбранную услугу в данный момент невозможна

      </p>

      <router-link *to*="/shedule/service" *class*="button-simple button-link"

        >К выбору услуг</router-link

      >

    </div>

    <div v-if="CURRENT\_MASTER">

      <label *for*="dateAppointment" *class*="shedule\_\_title"

        >Выберите дату:

      </label>

      <input *type*="date" *id*="dateAppointment" *v-model*.*lazy*="selectedDate" />

    </div>

    <div v-if="selectedDate && validDate" *class*="shedule\_\_select">

      <div *class*="shedule\_\_title">Выберите время для записи:</div>

      <div v-if="CURRENT\_MASTER" *class*="buttons-time-box">

        <button

          v-for="time in getFreeTime"

          :*key*="time.index"

          @*click*="selectedTime = time"

*class*="button-simple button-shedule button-time"

          :*class*="selectedTime === time ? 'active' : ''"

        >

          {{ printTime(time) }}

        </button>

      </div>

    </div>

    <div>

      <p v-if="getFreeTime?.length === 0" *class*="shedule\_\_message">

        Извините, к выбранному мастеру {{ selectedDate }} запись невозможна.

        Попробуйте выбрать другой день

      </p>

      <p v-if="!validDate" *class*="shedule\_\_message">

        Извините, на выбранную дату запись невозможна

      </p>

    </div>

    <div v-if="selectedTime">

      <span *class*="shedule\_\_title">Выбранная дата: </span

      ><span *class*="shedule\_\_value">{{ printDate(selectedDate) }}. </span>

      <span *class*="shedule\_\_title">Выбранное время: </span

      ><span *class*="shedule\_\_value">{{ printTime(selectedTime) }} </span>

    </div>

    <button

      v-if="selectedDate && selectedTime"

      :*to*="`/shedule/master/${CURRENT\_MASTER.id}/${

        CURRENT\_SERVICE.id

      }/${selectedDate}\_${printTime(selectedTime)}`"

*class*="button-simple button-link"

      @*click*="addNewShedule()"

    >

      Записаться

    </button>

  </div>

</template>

<script>

import { mapGetters, mapMutations } from "vuex";

import utils from "../../services/utils";

export default {

  name: "MakeAppointment",

  data() {

    return {

      selectedDate: null,

      selectedTime: null,

    };

  },

  mounted() {

    // выборка мастеров, которые могут оказать выбранную услугу, если мастер не выбран ранее

    if (!*this*.CURRENT\_MASTER) {

*this*.$store.dispatch(

        "GET\_MASTERS\_BY\_GROUP",

*this*.CURRENT\_SERVICE.groupServiceId

      );

    }

  },

  watch: {

    // при изменении даты, у сервера запрашивается расписание у выбранного мастера на выбранную дату

    selectedDate: function () {

      if (*this*.validDate) {

        const param = { date: *this*.selectedDate, id: *this*.CURRENT\_MASTER.id };

*this*.$store.dispatch("GET\_SHEDULE\_BY\_DATE\_AND\_MASTER", param);

      }

    },

  },

  computed: {

    ...mapGetters([

      "CURRENT\_MASTER",

      "CURRENT\_SERVICE",

      "MASTERS",

      "SHEDULE",

      "GET\_SERVICE\_FOR\_ID",

      "USER\_FOR\_SHEDULE",

    ]),

    currentUser() {

      return *this*.$store.state.auth.user;

    },

    validDate() {

      if (!*this*.selectedDate) {

        return true;

      }

      let select = **new** *Date*(`${*this*.selectedDate}`).getTime();

      let now = **new** *Date*().getTime();

      if (select > now) {

        return true;

      } else {

        return false;

      }

    },

    getFreeTime() {

      if (!*this*.selectedDate) {

        return null;

      }

      const freeTimeList = [];

      const busyTimeList = [];

      // Время начала и окончания работы мастеров

      let startTime = **new** *Date*(`${*this*.selectedDate} 10:00`);

      let finishTime = **new** *Date*(`${*this*.selectedDate} 20:00`);

      const selectServiceDuration = *this*.CURRENT\_SERVICE.duration / 30 - 1;

      // Формирование списка занятого времени у мастера, с учетом длительности услуг

*this*.SHEDULE.forEach((*el*) => {

        let busyTime = **new** *Date*(`${*el*.date} ${*el*.time}`);

        // длительность услуги, уже присутствующей в расписании (шагами в 30 минут)

        const findBusyTimeForService =

*this*.GET\_SERVICE\_FOR\_ID(*el*.serviceId).duration / 30;

        for (let i = 0; i < findBusyTimeForService; i++) {

          busyTimeList.push(busyTime.getTime());

          busyTime.setMinutes(busyTime.getMinutes() + 30);

        }

        // необходимо вычеркнуть длительность выбранной для записи услуги перед занятым временем мастера, чтобы не пересекалась новая и старая запись

        let busyTimeOfSelectServ = **new** *Date*(`${*el*.date} ${*el*.time}`);

        busyTimeOfSelectServ.setMinutes(busyTimeOfSelectServ.getMinutes() - 30);

        for (let i = 0; i < selectServiceDuration; i++) {

          busyTimeList.push(busyTimeOfSelectServ.getTime());

          busyTimeOfSelectServ.setMinutes(

            busyTimeOfSelectServ.getMinutes() - 30

          );

        }

      });

      // Формирование списка свободного времени для возможной записи

      while (startTime.getTime() < finishTime.getTime()) {

        freeTimeList.push(**new** *Date*(startTime.getTime()));

        startTime.setMinutes(startTime.getMinutes() + 30);

      }

      // Выбранная заказчиком услуга не должна заканчиваться после времени окончания работы салона

      finishTime.setMinutes(finishTime.getMinutes() - 30);

      for (let i = 0; i < selectServiceDuration; i++) {

        busyTimeList.push(finishTime.getTime());

        finishTime.setMinutes(finishTime.getMinutes() - 30);

      }

      // из времени работы салона freeTimeList вычеркиваются занятые слоты времени, полученные из busyTimeList

      return freeTimeList.filter((*el*) => !busyTimeList.includes(*el*.getTime()));

    },

    // проверка, если переход был с выбора услуги, то надо показывать панель выбора мастеров

    showSelectMaster() {

      const templatePath = `/shedule/service/${*this*.CURRENT\_SERVICE.id}`;

      return *this*.$route.path === templatePath;

    },

  },

  methods: {

    ...mapMutations(["SET\_MASTERS", "ADD\_NEW\_SHEDULE"]),

    printTime(*time*) {

      return utils.printTime(*time*);

    },

    printDate(*dateStr*) {

      return utils.printDate(*dateStr*);

    },

    getDate(*date*) {

      return utils.getDate(*date*);

    },

    addNewShedule() {

      const newShedule = {

        date: *this*.getDate(*this*.selectedTime),

        time: *this*.printTime(*this*.selectedTime),

        userId: *this*.USER\_FOR\_SHEDULE || *this*.currentUser.id,

        masterId: *this*.CURRENT\_MASTER.id,

        serviceId: *this*.CURRENT\_SERVICE.id,

      };

*this*.$store.dispatch("CREATE\_SHEDULE", newShedule).then((*res*)=>{

        console.log(*res*)

*this*.$router.push({

        path: `/shedule/recordSuccess/${*res*.date}\_${*res*.time}`,

      })

      }

      ).catch((*e*) => {

        console.log(*e*);

      })

    },

    setCurrentMaster(*idMaster*) {

*this*.$store.dispatch("GET\_MASTER\_BY\_ID", *idMaster*);

*this*.selectedDate = null;

    },

  },

};

</script>

<style *lang*="scss" *scoped*>

@import "../../styles/shedule";

*#dateAppointment* {

  font-size: 22px;

  font-family: "Cormorant Garamond";

}

*.subscribeBtn* {

  height: 60px;

  background-color: #cdaa7d;

  border-radius: 10px;

  color: #0a1111;

  border: 1px solid transparent;

  cursor: pointer;

  font-family: "Cormorant Garamond";

  font-size: 24px;

  font-style: normal;

  font-weight: 700;

  line-height: 60px;

  box-sizing: border-box;

  text-decoration: none;

  padding-left: 30px;

  padding-right: 30px;

  transition: all 0.3s;  
  &*:hover* {

    background-color: $color-hover;

    border: 1px solid #cdaa7d;

  }

}

*.shedule\_\_message* {

  margin-bottom: 30px;

  color: $color-hover;

}

*.button-shedule* {

  height: 40px;

  font-size: 22px;

  padding: 0 15px 0 15px;

}

*.buttons-time-box* {

  display: flex;

  flex-wrap: wrap;

  gap: 10px;

  justify-content: center;

}

*.button-time* {

  width: 90px;

}

</style>

**Приложение 46**

**Код компонента «SuccessfulAppointment.vue»**

<template>

  <div *class*="container shedule">

    <h3>Спасибо за запись!</h3>

    <p>

      Наш мастер

      <span *class*="shedule\_\_value">{{ CURRENT\_MASTER.name }}</span> ждет Вас в

      нашем салоне

    </p>

    <p *class*="shedule\_\_value shedule\_\_title">

      {{ printDate(selectedDate) }} в {{ selectedTime }}

    </p>

    <button @*click*="cleanCurrentValue()" *class*="subscribeBtn button-margin">

      Вернуться на главную

    </button>

  </div>

</template>

<script>

import { mapGetters, mapMutations } from "vuex";

import utils from "../../services/utils";

export default {

  name: "SuccessfulAppointment",

  data() {

    return {};

  },

  computed: {

    ...mapGetters([

      "CURRENT\_MASTER",

    ]),

    selectedDate() {

      return *this*.$route.params.recDate.slice(0, 10);

    },

    selectedTime() {

      return *this*.$route.params.recDate.slice(11, 16);

    },

  },

  methods: {

    ...mapMutations([

      "SET\_CURRENT\_SERVICE",

      "SET\_CURRENT\_MASTER",

      "SET\_SHEDULE",

      "SET\_USER\_FOR\_SHEDULE"

    ]),

    printDate(*date*){

      return utils.printDate(*date*);

    },

    cleanCurrentValue() {

*this*.SET\_CURRENT\_MASTER(null);

*this*.SET\_CURRENT\_SERVICE(null);

*this*.SET\_SHEDULE(null);

*this*.SET\_USER\_FOR\_SHEDULE(null);

*this*.$router.push({ path: `/main` });

    },

  },

};

</script>

<style *lang*="scss" *scoped*>

@import "../../styles/shedule";

*.subscribeBtn* {

  height: 60px;

  background-color: #cdaa7d;

  border-radius: 10px;

  color: #0a1111;

  border: 1px solid transparent;

  cursor: pointer;

  font-family: "Cormorant Garamond";

  font-size: 24px;

  font-style: normal;

  font-weight: 700;

  line-height: 60px;

  box-sizing: border-box;

  text-decoration: none;

  padding-left: 30px;

  padding-right: 30px;

  transition: all 0.3s;

  &*:hover* {

    background-color: white;

    border: 1px solid #cdaa7d;

  }

}

*.button-margin* {

  margin-top: 30px;

}

</style>

**Приложение 47**

**Код утилит «utils.js»**

class utils {

  printTime(dateTime) {

    return `${dateTime.getHours()}:${

      dateTime.getMinutes() < 10 ? 0 : ""

    }${dateTime.getMinutes()}`;

  }

  printDate(dateStr) {

    const year = *dateStr*.slice(0, 4);

    const day = *dateStr*.slice(8, 10);

    let month;

    switch (*dateStr*.slice(5, 7)) {

      case "01":

        month = "января";

        break;

      case "02":

        month = "февраля";

        break;

      case "03":

        month = "марта";

        break;

      case "04":

        month = "апреля";

        break;

      case "05":

        month = "мая";

        break;

      case "06":

        month = "июня";

        break;

      case "07":

        month = "июля";

        break;

      case "08":

        month = "августа";

        break;

      case "09":

        month = "сентября";

        break;

      case "10":

        month = "октября";

        break;

      case "11":

        month = "ноября";

        break;

      case "12":

        month = "декабря";

        break;

      default:

        break;

    }

    return `${day} ${month} ${year} года`;

  }

  getDate(*date*) {

    return `${*date*.getFullYear()}-${(*date*.getMonth() + 1) < 10 ? 0 : ""}${

*date*.getMonth() + 1

    }-${*date*.getDate() < 10 ? 0 : ""}${*date*.getDate()}`;

  }

}

export default **new** utils();