Code source - PWA

Jonathan Borel-Jaquet

10 juin 2021

Listings

Listing 1 – ./Sources/src/App.vue

```
<!--
  App.vue

  Main root component of the application also serving as a template for the
      application.

  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
-->

<template>
  <div id="app">
    <div id="content">
      <Navbar />
      <router-view />
    </div>
    <Footer />
  </div>
</template>

<script>
import Footer from "./components/Footer.vue";
import Navbar from "./components/Navbar.vue";

export default {
  name: "app",
  components: {
    Navbar,
    Footer,
  },
  created() {
    this.$store.dispatch("autoLogin");
  },
};
</script>

<style>
#content {
  min-height: calc(100vh - 260px);
}
</style>
```

Listing 2 – ./Sources/src/components/About.vue

```
<!--
  About.vue

  Component representing the page about of the application.

  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
-->

<template>
  <div>
    <left-section-content
      :textH1="texth1LeftSectionContent"
      :textH5="texth5LeftSectionContent"
      :textParagraph="textParagraphLeftSectionContent"
      :imageSource="imageSourceLeftSectionContent"
    ></left-section-content>
    <right-section-content
      :textH5="texth5RightSectionContent"
      :textParagraph="textParagraphRightSectionContent"
      :routeBouton="routeBoutonRightSectionContent"
      :imageSource="imageSourceRightSectionContent"
    ></right-section-content>
  </div>
</template>

<script>
import LeftSectionContent from "./LeftSectionContent.vue";
import BorisEducator from "../assets/img/boriseducateurcanin.png";
import RightSectionContent from "./RightSectionContent.vue";
import BorisWithHyron from "../assets/img/boriswithhyron.png";

export default {
  components: { LeftSectionContent, RightSectionContent },
  name: "About",
  data() {
    return {
      texth1LeftSectionContent: "Bonjour, je suis Boris Gourdoux",
      texth5LeftSectionContent: "+ de 300 couples Maître/chien à mon actif",
      textParagraphLeftSectionContent:
        "Depuis ma toute petite enfance, ma relation avec le chien à été salvatrice. Elle m’a permis de dépasser bien des difficultés et de trouver une véritable complicité. Diplômé et spécialisé en rééducation et comportement, vous bénéficiez de toute mon expertise et expérience dans l’éducation canine, fruit de nombreuses formations dans ce qui se fait de mieux dans le domaine. L’objectif chez Douceur de Chien est de vous permettre de créer un lien renforcé et durable avec votre compagnon à 4 pattes et ceci en utilisant uniquement de la patience, de la cohérence et beaucoup de passion. Ainsi chez Douceur de Chien vous n’obtiendrez aucun conseil sur l’utilisation de colliers étrangleurs ou même de la force pour éduquer votre chien.",
      imageSourceLeftSectionContent: BorisEducator,

      texth5RightSectionContent: "Une méthode moderne",
      textParagraphRightSectionContent:
        "Je base ma méthode en respectant les besoins et l’émotionnel du chien, c’est pourquoi je met l’accent sur une coopération totale de ce dernier en lui permettant de faire ses propres choix et ainsi vous permettre d’obtenir de votre chien, écoute et attention en toutes circonstances.",
      imageSourceRightSectionContent: BorisWithHyron,
      routeBoutonRightSectionContent: "about",
    };
  },
};
</script>

<style scoped></style>
```

Listing 3 – ./Sources/src/components/Administration.vue

```
<!--
  Administration.vue

  Component representing the administration page of the canine educator of the application.

  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
-->

<template>
  <div>
    <b-container>
      <b-row class="title text-center">
        <b-col>
          <h1>Administration</h1>
        </b-col>
      </b-row>
      <b-row>
        <b-col>
          <b-button
            id="btnUpdatePassword"
            variant="outline-primary"
            v-b-modal.modal-update-password
            block
          >
            Modifier mon mot de passe
          </b-button>
          <b-button
            id="btnAddClient"
            style="margin-bottom: 10px; margin-top: 10px"
            variant="outline-primary"
            block
            v-b-modal.modal-add-client
          >
            Ajouter un client
          </b-button>
        </b-col>
      </b-row>
      <b-row>
        <b-col>
          <b-input-group style="margin-bottom: 10px">
            <b-form-input
              id="filter-input"
              v-model="filter"
              type="search"
              placeholder="Tapez pour rechercher"
            ></b-form-input>

            <b-input-group-append>
              <b-button :disabled="!filter" @click="filter = ’’">
                Effacer
              </b-button>
            </b-input-group-append>
          </b-input-group>
        </b-col>
      </b-row>
      <b-pagination
        v-model="btableOptions.currentPage"
        align="fill"
        size="sm"
        class="my-0"
        :total-rows="totalRowsPagination"
      ></b-pagination>
      <b-table
```

```
        ref="tableuser"
        :items="btableOptions.items"
        :busy="isBusy"
        :fields="btableOptions.fields"
        :sort-by.sync="btableOptions.sortBy"
        :sort-desc.sync="btableOptions.sortDesc"
        :filter="filter"
        :filter-included-fields="filterOn"
        :perPage="10"
        :current-page="btableOptions.currentPage"
        sort-icon-left
        striped
        responsive="sm"
      >
        <template #table-busy>
          <div class="text-center text-danger my-2">
            <b-spinner class="align-middle"></b-spinner>
            <strong> Chargement...</strong>
          </div>
        </template>

        <template #cell(chiens)="row">
          <b-button size="sm" @click="row.toggleDetails" class="mr-2 tableBtn">
            {{ row.detailsShowing ? "Cacher" : "Afficher" }}
          </b-button>
        </template>
        <template #cell(edition)="row">
          <b-button
            :to="{
              name: ’customerInformation’,
              params: { userId: row.item.id },
            }"
            class="tableBtn"
          >
            <b-icon-pencil-square></b-icon-pencil-square>
          </b-button>
        </template>
        <template #row-details="row">
          <div v-if="row.item.dogs.length > 0">
            <b-card
              style="margin-bottom: 5px"
              v-for="dog in row.item.dogs"
              :key="dog.id"
            >
              <b-row>
                <b-col><b>Nom du chien:</b></b-col>
                <b-col>{{ dog.name }}</b-col>
              </b-row>
              <b-row>
                <b-col><b>Photo du chien:</b></b-col>
                <b-col v-if="dog.picture_serial_id">
                  <b-img
                    :src="
                      $API_URL + ’dogs/downloadPicture/’ + dog.picture_serial_id
                    "
                    rounded
                    fluid
                    :alt="dog.name + ’ picture’"
                  ></b-img>
                </b-col>
                <b-col v-else> Le chien n’a pas encore de photo</b-col>
              </b-row>
            </b-card>
          </div>
          <b-card v-else>
            <b-row>
              <b-col>L’utilisateur n’a aucun chiens pour l’instant</b-col>
            </b-row>
          </b-card>
        </template>
      </b-table>

      <!-- MODAL ADD CLIENT  -->
      <b-modal
        id="modal-add-client"
        title="Ajouter un client"
        :hide-footer="true"
      >
        <b-form
          @submit.prevent="
            createUser(
              formAddClient.email,
              formAddClient.firstname,
              formAddClient.lastname,
              formAddClient.phonenumber,
              formAddClient.address
            )
          "
        >
          <b-form-group
            id="input-group-client-email"
            label="Adresse e-mail :"
            label-for="input-client-email"
          >
            <b-form-input
              id="input-client-email"
              v-model="formAddClient.email"
              type="email"
              placeholder="Entrez l’adresse e-mail du client"
              required
            ></b-form-input>
          </b-form-group>
          <b-form-group
            id="input-group-client-firstname"
            label="Prénom :"
            label-for="input-client-firstname"
          >
            <b-form-input
              id="input-client-firstname"
              v-model="formAddClient.firstname"
              type="text"
              placeholder="Entrez le prénom du client"
              required
            ></b-form-input>
          </b-form-group>
          <b-form-group
            id="input-group-client-lastname"
            label="Nom de famille :"
            label-for="input-client-lastname"
          >
            <b-form-input
              id="input-client-lastname"
              v-model="formAddClient.lastname"
              type="text"
              placeholder="Entrez le nom de famille du client"
              required
            ></b-form-input>
          </b-form-group>
          <b-form-group
            id="input-group-client-phonenumber"
            label="Numéro de téléphone :"
            label-for="input-client-phonenumber"
          >
            <b-form-input
              id="input-client-phonenumber"
              v-model="formAddClient.phonenumber"
              type="text"
              placeholder="Entrez le numéro de téléphone du client"
              required
            ></b-form-input>
          </b-form-group>
          <b-form-group
            id="input-group-client-address"
            label="Adresse :"
            label-for="input-client-address"
          >
            <b-form-input
              id="input-client-address"
              v-model="formAddClient.address"
              type="text"
              placeholder="Entrez l’addresse du client"
              required
            ></b-form-input>
          </b-form-group>
          <b-button block type="submit" variant="outline-primary">
            Ajouter le client
          </b-button>
        </b-form>
      </b-modal>

      <!-- MODAL UPDATE PASSWORD -->
      <b-modal
        id="modal-update-password"
        title="Modifier le mot de passe"
        :hide-footer="true"
      >
        <b-form
          @submit.prevent="updateAuthCustomerPassword(password, repeatPassword)"
        >
          <b-form-group
            id="input-group-inscription-password"
            label="Mot de passe :"
            label-for="input-inscription-password"
          >
            <b-form-input
              id="input-inscription-password"
              v-model="formChangePassword.password"
              type="password"
              placeholder="Entrez le mot de passe"
              required
            ></b-form-input>
          </b-form-group>
          <b-form-group
            id="input-group-inscription-second-password"
            label="Confirmation du mot de passe :"
            label-for="input-inscription-second-password"
          >
            <b-form-input
              id="input-inscription-second-password"
              v-model="formChangePassword.repeatPassword"
              type="password"
              placeholder="Entrez le mot de passe"
              required
            ></b-form-input>
          </b-form-group>

          <b-button block type="submit" variant="outline-primary">
            Modifier le mot de passe
          </b-button>
        </b-form>
      </b-modal>
    </b-container>
  </div>
</template>

<script>
import { BIconPencilSquare } from "bootstrap-vue";

export default {
  components: {
    BIconPencilSquare,
  },
  name: "Administration",
  data() {
    return {
      btableOptions: {
        sortBy: "nom",
        sortDesc: false,
        fields: [
          { key: "nom", sortable: true },
          { key: "prenom", sortable: true },
          { key: "chiens", sortable: false },
          { key: "edition", sortable: false },
        ],
        items: [],
        currentPage: 1,
      },
      formAddClient: {
        email: "",
        firstname: "",
        lastname: "",
        phonenumber: "",
        address: "",
      },
      formChangePassword: {
        password: "",
        repeatPassword: "",
      },
      filter: null,
      filterOn: ["nom", "prenom"],
      isBusy: true,
      totalRowsPagination: 1,
    };
  },
  methods: {
    /**
     * Method to load all customers with their dogs from the api rest endpoint "GET api/v1/users".
     *
     */
    loadCustomersWithDogs() {
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token
        },
      };
      this.$http
        .get(this.$API_URL + "users/", config)
        .then((response) => {
          this.btableOptions.items = [];
          response.data.forEach((user) => {
            user["prenom"] = user["firstname"];
            delete user["firstname"];
            user["nom"] = user["lastname"];
            delete user["lastname"];
          });
          this.btableOptions.items = response.data;
          this.toggleBusy();
          this.totalRowsPagination = this.btableOptions.items.length + 10;
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method to create a user with the api rest endpoint "POST api/v1/users".
     *
     * @param {string} email The user's email address
     * @param {string} firstname The user's first name
     * @param {string} lastname The user's last name
     * @param {string} phonenumber The user's phonenumber
     * @param {string} address The user's address
     */
    createUser(email, firstname, lastname, phonenumber, address) {
      const params = new URLSearchParams();
      params.append("email", email);
      params.append("firstname", firstname);
      params.append("lastname", lastname);
      params.append("phonenumber", phonenumber);
      params.append("address", address);
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token,
        },
      };
      this.$http
        .post(this.$API_URL + "users/", params, config)
        .then((response) => {
          console.log(response);
          this.loadCustomersWithDogs();
          this.$alertify.success("Client ajouté avec succès");
          this.$bvModal.hide("modal-add-client");
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method to update the password of the authenticated user with the api rest endpoint "PATCH api/v1/users/me/changePassword".
     *
     * @param {string} password The new password
     * @param {string} repeatPassword Repetition of the new password
     */
    updateAuthCustomerPassword(password, repeatPassword) {
      if (password != repeatPassword) {
        this.$alertify.error("les mots de passes ne corréspondent pas");
        return;
      }
      const params = new URLSearchParams();
      params.append("password", password);
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token,
        },
      };
      this.$http
        .patch(this.$API_URL + "users/me/changePassword/", params, config)
        .then((response) => {
          console.log(response);
          this.$alertify.success("Mot de passe modifié");
          this.$bvModal.hide("modal-update-password");
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    toggleBusy() {
      this.isBusy = false;
    },
  },
  mounted() {
    this.loadCustomersWithDogs();
  },
};
</script>

<style scoped>
.tableBtn {
  background-color: #3ea3d8;
}
</style>
```

Listing 4 – ./Sources/src/components/ButtonReturn.vue

```
<!--
  ButtonReturn.vue

  Reusable component representing a static button on a page allowing to redirect the user to another component.

  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
-->

<template>
  <div>
    <b-button class="btnReturn" :to="to">
      <p class="h4">
        <b-icon-arrow-return-left></b-icon-arrow-return-left>
      </p>
    </b-button>
  </div>
</template>

<script>
import { BIconArrowReturnLeft } from "bootstrap-vue";

export default {
  components: {
    BIconArrowReturnLeft,
  },
  name: "ButtonReturn",
  props: {
    to: Object,
  },
};
</script>

<style scoped>
.btnReturn {
  padding-top: 18px;
  position: fixed;
  bottom: 100px;
  right: 15px;
  z-index: 5;
  display: block;
  height: 70px;
  width: 70px;
  border-radius: 50%;
  background-color: #008afc;
  box-shadow: -1px -1px 15px 1px rgba(0, 0, 0, 0.7);
}
</style>
```

Listing 5 – ./Sources/src/components/Calendar.vue

```
<!--
  Calendar.vue

  Component representing the page of the calendars of the dog educators of the society.

  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
```

```
-->

<template>
  <div>
    <b-container>
      <b-row id="title" class="text-center">
        <b-col>
          <h1>Agenda des éducateurs canins de la société</h1>
        </b-col>
      </b-row>
      <b-row id="title" class="text-center">
        <b-col>
          <b-form-group label="Éducateur canin :">
            <b-form-select
              id="educatorSelect"
              v-model="idEducatorSelected"
              @change="onChange()"
            >
              <b-form-select-option
                v-for="educator in educators"
                :key="educator.id"
                :value="educator.id"
                >{{ educator.firstname }}
                {{ educator.lastname }}</b-form-select-option
              >
            </b-form-select>
          </b-form-group>
        </b-col>
      </b-row>
      <b-row class="text-center">
        <b-col>
          <p class="font-weight-light">
            Vous voulez prendre un rendez-vous avec un des éducateurs canins de
            la société ? <br />
            Il vous suffit de cliquer sur un des créneaux horaires une fois
            connecté afin de réserver votre rendez-vous
          </p>
        </b-col>
      </b-row>
      <b-row class="text-center">
        <b-col>
          <b-badge id="badge-free" variant="success">Rendez-vous libre</b-badge>
        </b-col>
      </b-row>
      <b-row class="text-center">
        <b-col>
          <FullCalendar ref="fullCalendar" :options="calendarOptions" />
        </b-col>
      </b-row>
      <b-modal
        id="modal-create-appointment"
        title="Réserver un rendez-vous"
        :hide-footer="true"
      >
        <div v-if="auth">
          <b-form
            @submit.prevent="
              createAppointment(
                selectedAppointment.datetime,
                selectedAppointment.durationInHour,
                $store.state.user_id,
                selectedAppointment.idEducator
              )
            "
          >
            <b-row>
              <b-col class="d-flex justify-content-center">
                <h5>Éducateur canin</h5>
              </b-col>
            </b-row>
            <b-row>
              <b-col class="d-flex justify-content-center">
                <p>
                  {{ this.$jquery("#educatorSelect option:selected").text() }}
                </p>
              </b-col>
            </b-row>
            <b-row>
              <b-col class="d-flex justify-content-center">
                <h5>Date</h5>
              </b-col>
            </b-row>
            <b-row>
              <b-col>
                <b-calendar
                  block
                  id="date-calendar"
                  label-help=""
                  :readonly="true"
                  :value="selectedAppointment.datetime"
                ></b-calendar>
              </b-col>
            </b-row>
            <b-row>
              <b-col class="d-flex justify-content-center">
                <h5>Créneau horaire</h5>
              </b-col>
            </b-row>
            <b-row>
              <b-col class="d-flex justify-content-center">
                <p>
                  {{ selectedAppointment.startHour }}h à
                  {{ selectedAppointment.endHour }}h
                </p>
              </b-col>
            </b-row>
            <b-button block type="submit" variant="outline-primary">
              Réserver le rendez-vous
            </b-button>
          </b-form>
        </div>
        <div v-else>
          <p>Connectez-vous afin de pouvoir planifier votre rendez-vous.</p>
          <b-button to="/connection" variant="primary">Se connecter</b-button>
        </div>
      </b-modal>
    </b-container>
  </div>
```

```
</template>

<script>
import FullCalendar from "@fullcalendar/vue";
import dayGridPlugin from "@fullcalendar/daygrid";
import timeGridPlugin from "@fullcalendar/timegrid";
import interactionPlugin from "@fullcalendar/interaction";
import frLocale from "@fullcalendar/core/locales/fr";
import moment from "moment";
moment.locale("fr-ch");

export default {
  components: {
    FullCalendar,
  },
  name: "Calendar",
  data() {
    return {
      calendarOptions: {
        plugins: [dayGridPlugin, timeGridPlugin, interactionPlugin],
        initialView: "dayGridMonth",
        headerToolbar: {
          left: "prev,next today",
          center: "title",
          right: "dayGridMonth,timeGridWeek,timeGridDay",
        },
        height: "auto",
        locale: frLocale,
        eventDisplay: "block",
        allDaySlot: false,
        slotMinTime: "06:00:00",
        slotMaxTime: "20:00:00",
        events: [],
        eventBackgroundColor: "green",
        eventClick: this.handleDateClick,
      },
      selectedAppointment: {
        datetime: "",
        startHour: "",
        endHour: "",
        durationInHour: null,
        idEducator: null,
      },
      idEducatorSelected: null,
      educators: [],
    };
  },
  methods: {
    /**
     * Method to load all educators from the api rest endpoint "GET api/v1/users/educators".
     *
     */
    loadEducators() {
      this.$http.get(this.$API_URL + "users/educators/").then((response) => {
        this.educators = response.data;
        this.idEducatorSelected = this.educators[0].id;
        this.loadEducatorPlanning(this.idEducatorSelected);
      });
    },
    /**
     * Method to load all the time slots of an educator from his id from the api rest endpoint "GET api/v1/plannings/{idEducator}".
     *
     * @param {number} idEducator The dog educator's id
     */
    loadEducatorPlanning(idEducator) {
      this.$http
        .get(this.$API_URL + "plannings/" + idEducator)
        .then((response) => {
          this.calendarOptions.events = [];
          response.data.forEach((event) => {
            this.calendarOptions.events.push({
              title: "Disponible",
              start: event.date + " " + event.time_start,
              end: event.date + " " + event.time_end,
              extendedProps: {
                idEducator: idEducator,
              },
            });
          });
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method called when clicking on a time slot in the calendar to redirect the user to the daily overview and open the modal for creating appointments.
     *
     * @param {number} info The object of FullCalendar
     */
    handleDateClick: function (info) {
      {
        let calendarApi = this.$refs.fullCalendar.getApi();
        if (info.view.type == "dayGridMonth") {
          calendarApi.gotoDate(info.event.endStr);
          calendarApi.changeView("timeGridDay");
        }
        if (
          info.view.type == "timeGridWeek" ||
          info.view.type == "timeGridDay"
        ) {
          var startDate = moment(info.event.start);
          var endDate = moment(info.event.end);
          var datetime = startDate.format("YYYY-MM-DD HH:mm:ss");
          var startHour = startDate.format("HH");
          var endHour = endDate.format("HH");
          var durationDifference = moment.duration(
            moment(info.event.end).diff(moment(info.event.start))
          );
          var durationInHour = durationDifference.asHours();
          this.sendAppointmentInformations(
            datetime,
            startHour,
            endHour,
            durationInHour,
            info.event.extendedProps.idEducator
          );
          this.$bvModal.show("modal-create-appointment");
        }
      }
    },
    /**
     * Method to create an appointment with the api rest endpoint "POST api/v1/appointments".
     *
     * @param {string} datetime The datetime of the appointment
     * @param {number} durationInHour The duration in hours of the appointment
     * @param {number} user_id_customer The customer's identifier
     * @param {number} user_id_educator The dog educator's identifier
     */
    createAppointment(
      datetime,
      durationInHour,
      user_id_customer,
      user_id_educator
    ) {
      const params = new URLSearchParams();
      params.append("datetime_appointment", datetime);
      params.append("duration_in_hour", durationInHour);
      params.append("user_id_customer", user_id_customer);
      params.append("user_id_educator", user_id_educator);
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token,
        },
      };
      this.$http
        .post(this.$API_URL + "appointments/", params, config)
        .then((response) => {
          console.log(response);
          this.$alertify.success("Rendez-vous reservé avec succès");
          this.$bvModal.hide("modal-create-appointment");
          this.loadEducatorPlanning(this.idEducatorSelected);
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method to load another calendar when changing the selection of a dog educator
     *
     */
    onChange() {
      this.loadEducatorPlanning(this.idEducatorSelected);
    },
    sendAppointmentInformations(
      datetime,
      startHour,
      endHour,
      durationInHour,
      idEducator
    ) {
      this.selectedAppointment.datetime = datetime;
      this.selectedAppointment.startHour = startHour;
      this.selectedAppointment.endHour = endHour;
      this.selectedAppointment.durationInHour = durationInHour;
      this.selectedAppointment.idEducator = idEducator;
    },
  },
  mounted() {
    this.loadEducators();
  },
  computed: {
    auth() {
      return this.$store.getters.ifAuthenticated;
    },
  },
};
</script>

<style scoped>
#badge-free {
  padding: 10px;
  margin-bottom: 10px;
  background-color: #008000;
  border: #3788d8 1px solid;
}
#title {
  margin-top: 20px;
  color: #3ea3d8;
}
</style>
```

Listing 6 – ./Sources/src/components/Connection.vue

```
<!--
  Connection.vue

  Component representing the application connection page.

  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
-->

<template>
  <div class="content">
    <b-container>
      <b-row id="title" class="text-center">
        <b-col>
          <h1>Se connecter</h1>
        </b-col>
      </b-row>
      <b-row class="text-center text-secondary">
        <b-col>
          Nouveau sur ce site ?
          <router-link to="/inscription">
            <span role="link">S'inscrire</span>
          </router-link>
        </b-col>
      </b-row>
      <b-row id="title" class="justify-content-center">
        <b-col lg="6">
          <b-form @submit.prevent="connection" autocomplete="on">
            <b-form-group
```

```
              id="input-group-connection-email"
              label="Adresse e-mail:"
              label-for="input-connection-email"
            >
              <b-form-input
                id="input-connection-email"
                v-model="form.email"
                type="email"
                placeholder="Entrez l'adresse e-mail"
                autocomplete="email"
                required
              ></b-form-input>
            </b-form-group>
            <b-form-group
              id="input-group-connection-password"
              label="Mot de passe :"
              label-for="input-connection-password"
            >
              <b-form-input
                id="input-connection-password"
                v-model="form.password"
                type="password"
                placeholder="Entrez le mot de passe"
                autocomplete="current-password"
                required
              ></b-form-input>
            </b-form-group>
            <b-button block type="submit" variant="outline-primary">
              Connexion
            </b-button>
          </b-form>
        </b-col>
      </b-row>
    </b-container>
  </div>
</template>

<script>
export default {
  name: "Connection",
  data() {
    return {
      form: {
        email: "",
        password: "",
      },
    };
  },
  methods: {
    /**
     * Method to connect the user with the store.
     *
     */
    connection() {
      this.$store.dispatch("login", {
        email: this.form.email,
        password: this.form.password,
      });
    },
  },
};
</script>

<style scoped>
#title {
  margin-top: 20px;
  color: #3ea3d8;
}
</style>
```

Listing 7 – ./Sources/src/components/CustomerAppointment.vue

```
<!--
  CustomerAppointment.vue

  Component representing the appointment information of a client.

  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
-->

<template>
  <div class="ComponnentContent">
    <b-container>
      <b-row class="text-center">
        <b-col>
          <h1>{{ title }} {{ customer.firstname }} {{ customer.lastname }}</h1>
        </b-col>
      </b-row>
      <b-row v-if="appointments.length > 0" class="content">
        <b-col>
          <div class="card">
            <div class="card-body">
              <div id="content">
                <ul class="timeline">
                  <li
                    class="event"
                    v-for="appointment in appointments"
                    :key="appointment.id"
                    :data-date="
                      appointment.hourStart + ’ - ’ + appointment.hourEnd
                    "
                  >
                    <h4>{{ appointment.date }}</h4>
                    <h3>Éducateur canin</h3>
                    <p>{{ appointment.educatorFullname }}</p>
                    <h3>Résumé du rendez-vous</h3>
                    <p v-if="appointment.summary != null">
                      {{ appointment.summary }}
                    </p>
                    <p v-else>Aucun résumé</p>
                    <div v-if="authAdministrator">
                      <div style="position: absolute; top: 5px; right: 5px">
                        <b-button
                          variant="danger"
                          v-b-modal.modal-delete-appointment
                          @click="
                            sendAppointmentInformations(
                              appointment.id,
                              appointment.summary,
                              appointment.noteText,
                              appointment.noteGraphicalSerialId
                            )
                          "
                        >
                          <b-icon-trash id="lol"></b-icon-trash>
                        </b-button>
                      </div>
                      <h3>Notes textuelles personnelles du rendez-vous</h3>
                      <p v-if="appointment.noteText != null">
                        {{ appointment.noteText }}
                      </p>
                      <p v-else>Aucunes notes textuelles</p>
                      <b-button
                        id="btnGraphicalNote"
                        variant="outline-primary"
                        block
                        v-b-modal.modal-graphical-note
                        @click="
                          sendAppointmentInformations(
                            appointment.id,
                            appointment.summary,
                            appointment.noteText,
                            appointment.noteGraphicalSerialId
                          )
                        "
                      >
                        Notes graphiques
                      </b-button>
                      <b-button
                        id="btnTextualNote"
                        variant="outline-primary"
                        block
                        v-b-modal.modal-update-appointment-informations
                        @click="
                          sendAppointmentInformations(
                            appointment.id,
                            appointment.summary,
                            appointment.noteText,
                            appointment.noteGraphicalSerialId
                          )
                        "
                      >
                        Notes textuelles
                      </b-button>
                    </div>
                  </li>
                </ul>
              </div>
            </div>
          </div>
        </b-col>
      </b-row>
      <b-row v-else class="content">
        <b-col>
          {{ messageNoAppointments }}
        </b-col>
      </b-row>
      <button-return
        v-if="authAdministrator"
        :to="{
          name: 'customerInformation',
          params: { userId: $route.params.userId },
        }"
      ></button-return>

      <!-- MODAL UPDATE APPOINTMENT INFORMATIONS-->
      <b-modal
        id="modal-update-appointment-informations"
        title="Modifier le notes du rendez-vous"
        :hide-footer="true"
      >
        <b-form
          @submit.prevent="
            updateAppointmentByAppointmentId(
              selectedAppointment.id,
              selectedAppointment.summary,
              selectedAppointment.noteText
            )
          "
        >
          <b-form-group
            id="input-group-appointment-summary"
            label="Résumé :"
            label-for="input-appointment-summary"
          >
            <b-form-textarea
              id="input-appointment-summary"
              v-model="selectedAppointment.summary"
              placeholder="Entrer le résumé du cours"
              rows="8"
              max-rows="8"
            ></b-form-textarea>
          </b-form-group>

          <b-form-group
            id="input-group-appointment-note-text"
            label="Notes textuelles personnelles :"
            label-for="input-appointment-note-text"
          >
            <b-form-textarea
              id="input-appointment-note-text"
              v-model="selectedAppointment.noteText"
              placeholder="Entrer les notes textuelles du cours"
              rows="8"
              max-rows="8"
            ></b-form-textarea>
          </b-form-group>

          <b-button block type="submit" variant="outline-primary">
            Modifier les notes
          </b-button>
        </b-form>
      </b-modal>

      <!-- MODAL GRAPHICAL NOTE-->
      <b-modal
```

```
        id="modal-graphical-note"
        modal-class="modal-fullscreen"
        title="Notes graphiques"
        :hide-footer="true"
        @shown="
          $refs.signaturePad.resizeCanvas();
          loadNoteGraphicalPicture(selectedAppointment.noteGraphicalSerialId);
        "
      >
        <b-form
          style="width: 100%"
          @submit.prevent="
            uploadNoteGraphicalByAppointmentId(
              selectedAppointment.id,
              $refs.signaturePad.saveSignature(’image/jpeg’).data
            )
          "
        >
          <VueSignaturePad
            id="signature"
            width="100%"
            height="80%"
            :customStyle="customStyleSignaturePad"
            ref="signaturePad"
            :options="{ backgroundColor: ’rgb(255, 255, 255)’ }"
          />
          <b-row style="margin-top: 10px">
            <b-col>
              <b-button
                block
                @click="clearNoteGraphical"
                variant="outline-danger"
              >
                Effacer
              </b-button>
            </b-col>
            <b-col>
              <b-button block type="submit" variant="outline-success">
                Sauvegarder
              </b-button>
            </b-col>
          </b-row>
        </b-form>
      </b-modal>

      <!-- MODAL DELETE APPOINTMENT-->
      <b-modal
        id="modal-delete-appointment"
        title="Confirmation de supression d’un rendez-vous"
        :hide-footer="true"
        :hide-header="true"
      >
        <h5 style="text-align: center">Supprimer le rendez-vous ?</h5>
        <b-form @submit.prevent="deleteAppointmentById(selectedAppointment.id)">
          <b-row>
            <b-col>
              <b-button
                block
                variant="danger"
                @click="$bvModal.hide(’modal-delete-appointment’)"
              >
                Non
              </b-button>
            </b-col>
            <b-col>
              <b-button
                block
                type="submit"
                variant="success"
                @click="$bvModal.hide(’modal-delete-appointment’)"
              >
                Oui
              </b-button>
            </b-col>
          </b-row>
        </b-form>
      </b-modal>
    </b-container>
  </div>
</template>

<script>
import ButtonReturn from "./ButtonReturn.vue";
import "@/assets/css/timeline.css";
import { BIconTrash } from "bootstrap-vue";
export default {
  name: "CustomerAppointment",
  components: {
    ButtonReturn,
    BIconTrash,
  },
  data() {
    return {
      title: "",
      customer: {
        firstname: "",
        lastname: "",
      },
      selectedAppointment: {
        id: null,
        summary: "",
        noteText: "",
        noteGraphicalSerialId: "",
      },
      messageNoAppointments: "Aucun rendez-vous",
      customStyleSignaturePad: { border: "#c2c2c2 3px solid" },
      appointments: [],
      educators: [],
    };
  },
  methods: {
    /**
     * Method to load all educators from the api rest endpoint "GET api/v1/users/educators".
     *
     */
    loadEducators() {
      this.$http.get(this.$API_URL + "users/educators/").then((response) => {
        this.educators = response.data;
        if (this.authAdministrator && this.$route.params.userId) {
          this.title = "Rendez-vous du client";
          this.loadCustomerAppointmentsByUserId(this.$route.params.userId);
        } else {
          this.title = "Mes rendez-vous";
          this.loadAuthCustomerAppointment();
        }
      });
    },
    /**
     * Method to load all appointments of the authenticated client from the api rest endpoint "GET api/v1/appointments".
     *
     */
    loadAuthCustomerAppointment() {
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token
        },
      };
      this.$http
        .get(this.$API_URL + "appointments/", config)
        .then((response) => {
          this.loadAppointmentsData(response.data);
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method to load all the appointments of a client from his id from the api rest endpoint "GET api/v1/users/{userId}"
     *
     * @param {number} userId The client's id
     */
    loadCustomerAppointmentsByUserId(userId) {
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token
        },
      };
      this.$http
        .get(this.$API_URL + "users/" + userId, config)
        .then((response) => {
          this.customer.firstname = response.data.firstname;
          this.customer.lastname = response.data.lastname;

          this.loadAppointmentsData(response.data.appointments);
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method for formatting the time data of an appointment object table.
     *
     * @param {Object[]} appointments The appointment object table
     */
```

```
  loadAppointmentsData(appointments) {
    this.appointments = [];
    appointments.forEach((appointment) => {
      var date = new Date(appointment.datetime_appointment.replace(" ", "T"));
      var options = {
        weekday: "long",
        year: "numeric",
        month: "long",
        day: "numeric",
      };
      var educator = this.educators.find(
        (x) => x.id === appointment.user_id_educator
      );
      var educatorFullname = educator.firstname + " " + educator.lastname;
      this.appointments.push({
        id: appointment.id,
        hourStart: date.getHours() + "h",
        hourEnd: date.getHours() + appointment.duration_in_hour + "h",
        date: date.toLocaleDateString("fr-CH", options),
        summary: appointment.summary,
        noteText: appointment.note_text,
        noteGraphicalSerialId: appointment.note_graphical_serial_id,
        educatorFullname: educatorFullname,
      });
    });
  },
  /**
   * Method to load a graphical note from its serial id from the api rest endpoint "GET api/v1/appointments/downloadNoteGraphical/{noteGraphicalSerialId}".
   *
   * @param {string} noteGraphicalSerialId The serial id of the graphical  note
   */
  loadNoteGraphicalPicture(noteGraphicalSerialId) {
    const config = {
      headers: {
        "Authorization" : this.$store.state.api_token
      },
      responseType: "arraybuffer",
    };
    this.$http
      .get(
        this.$API_URL +
          "appointments/downloadNoteGraphical/" +
          noteGraphicalSerialId,
        config
      )
      .then((response) => {
        var base64 =
          "data:image/png;base64," +
          Buffer.from(response.data, "binary").toString("base64");
        this.$refs.signaturePad.fromDataURL(base64);
      })
      .catch((error) => {
        this.$alertify.error(error.response.data.error);
      });
  },
  /**
   * Method to update an appointment from his id from the api rest endpoint "
```

```
        PATCH api/v1/appointments/{appointmentId}"
     *
     * @param {number} id The appointment id
     * @param {string} summary The summary of the appointment
     * @param {string} noteText The note text of the appointment
     */
    updateAppointmentByAppointmentId(id, summary, noteText) {
      const params = new URLSearchParams();
      params.append("summary", summary);
      params.append("note_text", noteText);
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token,
        },
      };
      this.$http
        .patch(this.$API_URL + "appointments/" + id, params, config)
        .then((response) => {
          console.log(response);
          this.loadCustomerAppointmentsByUserId(this.$route.params.userId);
          this.$alertify.success("Notes textuelles modifiées");
          this.$bvModal.hide("modal-update-appointment-informations");
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method to allow the component to read the data of an appointment in order to proceed to a modification or a deletion from a modal.
     *
     * @param {number} id The appointment id
     * @param {string} summary The summary of the appointment
     * @param {string} noteText The note text of the appointment
     * @param {string} noteGraphicalSerialId The serial id of the graphical note of the appointment
     */
    sendAppointmentInformations(id, summary, noteText, noteGraphicalSerialId) {
      this.selectedAppointment.id = id;
      this.selectedAppointment.summary = summary;
      this.selectedAppointment.noteText = noteText;
      this.selectedAppointment.noteGraphicalSerialId = noteGraphicalSerialId;
    },
    /**
     * Method to upload a graphical note from the api rest endpoint "POST api/v1/appointments/uploadNoteGraphical".
     *
     * @param {number} id The appointment id
     * @param {string} noteGraphicalBase64 The graphical note in base64 format
     */
    uploadNoteGraphicalByAppointmentId(id, noteGraphicalBase64) {
      //Workaround with the fetch api to convert a base64 to a Binary Large object
      fetch(noteGraphicalBase64)
        .then((res) => res.blob())
        .then((blob) => {
          var file = new File([blob], "note", { type: "image/jpeg" });
          let formData = new FormData();
          formData.append("note_graphical", file);
          formData.append("appointment_id", id);
          const config = {
            headers: {
              "Authorization" : this.$store.state.api_token
            },
          };
          this.$http
            .post(
              this.$API_URL + "appointments/uploadNoteGraphical/",
              formData,
              config
            )
            .then((response) => {
              console.log(response);
              this.loadCustomerAppointmentsByUserId(this.$route.params.userId);
              this.$alertify.success("Notes graphiques ajoutées avec succès");
              this.$bvModal.hide("modal-graphical-note");
            })
            .catch((error) => {
              this.$alertify.error(error.response.data.error);
            });
        });
    },
    /**
     * Method to delete an appointment from his id from the api rest endpoint "DELETE api/v1/appointments/{appointmentId}"
     *
     * @param {number} id The appointment id
     */
    deleteAppointmentById(id) {
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token,
        },
      };
      this.$http
        .delete(this.$API_URL + "appointments/" + id, config)
        .then((response) => {
          this.loadCustomerAppointmentsByUserId(this.$route.params.userId);
          this.$alertify.success("Rendez-vous supprimé");
          console.log(response);
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method for clearing the signature of the signaturePad component
     *
     */
    clearNoteGraphical() {
      this.$refs.signaturePad.clearSignature();
    },
  },
  computed: {
    authAdministrator() {
      return this.$store.getters.ifAdministratorAuthenticated;
    },
  },
  mounted() {
    this.loadEducators();
  },
};
</script>

<style scoped>
.ComponnentContent {
  margin-top: 20px;
  color: #3ea3d8;
}
</style>
```

Listing 8 – ./Sources/src/components/CustomerInformation.vue

```
<!--
  CustomerInformation.vue

  Component representing the personal information page of a client.

  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
-->

<template>
  <div>
    <b-container>
      <b-row class="text-center content">
        <b-col>
          <h1>{{ title }}</h1>
        </b-col>
      </b-row>
      <b-row class="content">
        <b-col
          md="4"
          style="margin-bottom: 10px"
          class="text-center d-flex flex-column align-items-center"
        >
          <b-card
            style="padding: 10px; width: 100%; margin-bottom: 10px"
            img-src="./../assets/img/user-profile.png"
            img-top
            :title="customer.firstname"
          >
            <p class="text-secondary mb-1" v-if="customer.code_role == ’1’">
              Client
            </p>
            <p class="text-secondary mb-1">{{ customer.address }}</p>
            <div v-if="authCustomer">
              <b-button
                id="btnUpdatePassword"
                variant="outline-primary"
                v-b-modal.modal-update-password
                block
              >
                Modifier mon mot de passe
              </b-button>
            </div>
            <div v-if="authAdministrator">
              <b-button
                id="btnGoToAppointment"
                :to="{
                  name: 'customerAppointment',
                  params: { userId: $route.params.userId },
                }"
                variant="outline-primary"
                class="btnAdmin"
                v-if="authAdministrator"
                >Rendez-vous
              </b-button>
              <b-button
                id="btnAddDog"
                variant="outline-primary"
                class="btnAdmin"
                v-if="authAdministrator"
                v-b-modal.modal-add-dog
              >
                Ajouter un chien
              </b-button>
              <b-button
                id="btnAddDocument"
                variant="outline-primary"
                class="btnAdmin"
                v-if="authAdministrator"
                v-b-modal.modal-add-document
              >
                Ajouter un document
              </b-button>
              <b-button
                id="btnDeleteUser"
                variant="outline-danger"
                class="btnAdmin"
                v-if="authAdministrator"
                v-b-modal.modal-delete-user
              >
                Supprimer l'utilisateur
              </b-button>
            </div>
          </b-card>

          <b-card
            v-if="customer.documents.length > 0"
            style="padding: 10px; width: 100%"
          >
            <div v-for="document in customer.documents" :key="document.id">
              <b-row style="text-align: center; margin-top: 10px">
                <b-col>
                  <b-button
                    @click="
                      downloadDocument(
                        document.type,
                        document.document_serial_id
                      )
                    "
                    class="btnAdmin"
                    block
                    variant="outline-primary"
                  >
                    {{ document.type }}_{{ document.document_serial_id }}
                    <b-icon-download></b-icon-download>
                  </b-button>
                  <b-button
                    class="btnAdmin"
                    block
                    variant="outline-danger"
                    v-if="authAdministrator"
                    v-b-modal.modal-delete-document
                    @click="sendDocumentId(document.id)"
                  >
                    Supprimer le document
                  </b-button>
                </b-col>
              </b-row>
              <hr style="margin-bottom: 0px; margin-top: 0px" />
            </div>
          </b-card>
        </b-col>
        <b-col md="8">
          <b-card style="margin-bottom: 10px">
            <b-row>
              <b-col sm="3">
                <h6 class="mb-0">Nom complet</h6>
              </b-col>
              <b-col sm="9" class="text-secondary">
                {{ customer.firstname }} {{ customer.lastname }}
              </b-col>
            </b-row>
            <hr />
            <b-row>
              <b-col sm="3">
                <h6 class="mb-0">Adresse e-mail</h6>
              </b-col>
              <b-col sm="9" class="text-secondary">{{ customer.email }}</b-col>
            </b-row>
            <hr />
            <b-row>
              <b-col sm="3">
                <h6 class="mb-0">Numéro de téléphone</h6>
              </b-col>
              <b-col sm="9" class="text-secondary">{{
                customer.phonenumber
              }}</b-col>
            </b-row>
            <hr />
            <b-row>
              <b-col sm="3">
                <h6 class="mb-0">Adresse de domicile</h6>
              </b-col>
              <b-col sm="9" class="text-secondary">{{
                customer.address
              }}</b-col>
            </b-row>
            <div v-if="authAdministrator">
              <hr />
              <b-button
                id="btnUpdateUser"
                variant="outline-primary"
                class="btnAdmin"
                v-b-modal.modal-update-user
              >
                Modifier l’utilisateur
              </b-button>
            </div>
          </b-card>
          <b-row>
            <b-col md="6" v-for="dog in customer.dogs" :key="dog.id">
              <b-card style="margin-bottom: 10px">
                <b-row no-gutters>
                  <b-col md="12">
                    <b-card-img
                      v-if="dog.picture_serial_id"
                      :src="
                        $API_URL +
                        ’dogs/downloadPicture/’ +
                        dog.picture_serial_id
                      "
                      alt="Image"
                      style="margin-bottom: 15px"
                      bottom
                    ></b-card-img>
                    <b-card-img
                      v-else
                      :src="require(’../assets/img/placeholder-dog.png’)"
                      alt="placeholder"
                      style="margin-bottom: 15px"
                    ></b-card-img>
                  </b-col>
                </b-row>
                <b-row>
                  <b-col sm="4">
                    <h6 class="mb-0">Nom</h6>
                  </b-col>
                  <b-col sm="8" class="text-secondary">{{ dog.name }}</b-col>
                </b-row>
                <hr />
                <b-row>
                  <b-col sm="4">
                    <h6 class="mb-0">Race</h6>
                  </b-col>
                  <b-col sm="8" class="text-secondary">{{ dog.breed }}</b-col>
                </b-row>
                <hr />
                <b-row>
                  <b-col sm="4">
                    <h6 class="mb-0">Sexe</h6>
                  </b-col>
                  <b-col sm="8" class="text-secondary">{{ dog.sex }}</b-col>
                </b-row>
                <hr />
                <b-row>
                  <b-col sm="4">
                    <h6 class="mb-0">Numéro de puce</h6>
                  </b-col>
                  <b-col sm="8" class="text-secondary">
                    <span v-if="dog.chip_id != ’’">{{ dog.chip_id }}</span>
                    <span v-else>-</span>
                  </b-col>
                </b-row>
                <div v-if="authAdministrator">
                  <hr />
                  <b-button
                    id="btnUpdateDog"
                    variant="outline-primary"
                    class="btnAdmin"
                    v-b-modal.modal-update-dog
                    @click="
                      sendDogId(dog.id);
                      sendDogInformations(
                        dog.name,
                        dog.breed,
                        dog.sex,
                        dog.chip_id
                      );
                    "
                  >
                    Modifier le chien
                  </b-button>
                  <b-button
                    id="btnAddPictureDog"
                    class="btnAdmin"
                    variant="outline-primary"
                    v-b-modal.modal-add-picture-dog
                    @click="sendDogId(dog.id)"
                  >
                    Modifier la photo
                  </b-button>
                  <b-button
                    id="btnDeleteDog"
                    variant="outline-danger"
                    class="btnAdmin"
                    v-b-modal.modal-delete-dog
                    @click="sendDogId(dog.id)"
                  >
                    Supprimer le chien
                  </b-button>
                </div>
              </b-card>
            </b-col>
          </b-row>
        </b-col>
      </b-row>
      <button-return
        v-if="authAdministrator"
        :to="{
          name: 'administration',
        }"
      ></button-return>

      <!-- MODAL ADD DOG-->
      <b-modal id="modal-add-dog" title="Ajouter un chien" :hide-footer="true">
        <b-form
          @submit.prevent="
            createDogForCustomerByUserId(
              addDogForm.name,
              addDogForm.breed,
              addDogForm.sex,
              addDogForm.chip_id,
              $route.params.userId
            )
          "
        >
          <b-form-group
            id="input-group-dog-name"
            label="Nom :"
            label-for="input-dog-name"
          >
            <b-form-input
              id="input-dog-name"
              v-model="addDogForm.name"
              type="text"
              placeholder="Entrez le nom du chien"
              required
            ></b-form-input>
          </b-form-group>

          <b-form-group
            id="input-group-dog-breed"
            label="Race :"
            label-for="input-dog-breed"
          >
            <b-form-input
              id="input-dog-breed"
              v-model="addDogForm.breed"
              type="text"
              placeholder="Entrez la race du chien"
              required
            ></b-form-input>
          </b-form-group>

          <b-form-group
            id="input-group-dog-sex"
            label="Sexe :"
            label-for="input-dog-sex"
          >
            <b-form-select
              id="input-dog-sex"
              v-model="addDogForm.sex"
              :options="sexDogOptions"
              required
            ></b-form-select>
          </b-form-group>

          <b-form-group
            id="input-group-dog-chip-id"
            label="Numéro de puce :"
            label-for="input-dog-chip-id"
          >
            <b-form-input
              id="input-dog-chip-id"
              v-model="addDogForm.chip_id"
              type="text"
              placeholder="Entrez le numéro de puce du chien"
            ></b-form-input>
          </b-form-group>

          <b-button block type="submit" variant="outline-primary">
```

```
            Ajouter le chien
          </b-button>
        </b-form>
      </b-modal>

      <!-- MODAL ADD DOG PICTURE-->
      <b-modal
        id="modal-add-picture-dog"
        title="Ajouter une photo de chien"
        :hide-footer="true"
      >
        <b-form
          @submit.prevent="
            uploadDogPictureByDogId(selectedDog.id, selectedDog.pictureFile)
          "
        >
          <b-form-group
            id="input-group-dog-picture-file"
            label="Photo :"
            label-for="input-dog-picture-file"
          >
            <b-form-file
              required
              v-model="selectedDog.pictureFile"
              accept="image/jpeg, image/png"
              placeholder="Aucun fichier choisi"
              browse-text="Ajouter"
              id="input-dog-picture-file"
              size="sm"
            ></b-form-file>
          </b-form-group>

          <b-button block type="submit" variant="outline-primary">
            Ajouter la photo
          </b-button>
        </b-form>
      </b-modal>

      <!-- MODAL ADD DOCUMENT-->
      <b-modal
        id="modal-add-document"
        title="Ajouter un document"
        :hide-footer="true"
        @hidden="resetModalAddDocument"
      >
        <b-form
          @submit.prevent="
            createDocumentForCustomerByUserId(
              $route.params.userId,
              addDocumentForm.type,
              signatureBase64,
              addDocumentForm.packageNumber,
              addDocumentForm.file
            )
          "
        >
          <b-form-group
            id="input-group-document-type"
            label="Type du document :"
            label-for="input-document-type"
          >
            <b-form-select
              id="input-document-type"
              v-model="addDocumentForm.type"
              :options="documentOptions"
              required
            ></b-form-select>
          </b-form-group>

          <div v-if="addDocumentForm.type == ’conditions_inscription’">
            <a href="./pdf/conditionsInscription.pdf" target="_blank">
              Conditions d’inscription
            </a>
            <b-form-group
              id="input-group-document-package-number"
              label="Forfait :"
              label-for="input-document-package-number"
            >
              <b-form-select
                id="input-document-package-number"
                v-model="addDocumentForm.packageNumber"
                :options="packageNumbers"
                required
              ></b-form-select>
            </b-form-group>
            <signature-pad @getSignaturePad="saveSignature"></signature-pad>
            <b-form-checkbox
              id="checkbox-validation"
              v-model="checkboxValidationStatus"
              name="checkbox-validation"
              @change="refreshStateOfcreateConditionsButton()"
              style="margin: 10px"
            >
              <h5>Lu et approuvé</h5>
            </b-form-checkbox>
            <b-button
              :disabled="createConditionsButtonisDisabled"
              block
              type="submit"
              variant="outline-primary"
            >
              Créer les conditions d’inscription
            </b-button>
          </div>
          <div v-else>
            <b-form-group
              id="input-group-document-file"
              label="Document PDF :"
              label-for="input-dog-document-file"
            >
              <b-form-file
                required
                v-model="addDocumentForm.file"
                placeholder="Aucun fichier choisi"
                browse-text="Ajouter"
                id="input-document-file"
                size="sm"
              ></b-form-file>
            </b-form-group>
            <b-button block type="submit" variant="outline-primary">
              Ajouter le document
            </b-button>
          </div>
        </b-form>
      </b-modal>

      <!-- MODAL DELETE USER-->
      <b-modal
        id="modal-delete-user"
        title="Confirmation de supression d’utilisateur"
        :hide-footer="true"
        :hide-header="true"
      >
        <h5 style="text-align: center">Supprimer l’utilisateur ?</h5>
        <b-form @submit.prevent="deleteUserById($route.params.userId)">
          <b-row>
            <b-col>
              <b-button
                block
                variant="danger"
                @click="$bvModal.hide(’modal-delete-user’)"
              >
                Non
              </b-button>
            </b-col>
            <b-col>
              <b-button block type="submit" variant="success">Oui</b-button>
            </b-col>
          </b-row>
        </b-form>
      </b-modal>

      <!-- MODAL DELETE DOCUMENT-->
      <b-modal
        id="modal-delete-document"
        title="Confirmation de supression d’utilisateur"
        :hide-footer="true"
        :hide-header="true"
      >
        <h5 style="text-align: center">Supprimer le document ?</h5>
        <b-form @submit.prevent="deleteDocumentById(selectedDocument.id)">
          <b-row>
            <b-col>
              <b-button
                block
                variant="danger"
                @click="$bvModal.hide(’modal-delete-document’)"
              >
                Non
              </b-button>
            </b-col>
            <b-col>
              <b-button
                block
                type="submit"
                variant="success"
                @click="$bvModal.hide(’modal-delete-document’)"
              >
                Oui
              </b-button>
            </b-col>
          </b-row>
        </b-form>
      </b-modal>

      <!-- MODAL DELETE DOG-->
      <b-modal
        id="modal-delete-dog"
        title="Confirmation de supression d’utilisateur"
        :hide-footer="true"
        :hide-header="true"
      >
        <h5 style="text-align: center">Supprimer le chien ?</h5>
        <b-form @submit.prevent="deleteDogById(selectedDog.id)">
          <b-row>
            <b-col>
              <b-button
                block
                variant="danger"
                @click="$bvModal.hide('modal-delete-dog')"
              >
                Non
              </b-button>
            </b-col>
            <b-col>
              <b-button
                @click="$bvModal.hide('modal-delete-dog')"
                block
                type="submit"
                variant="success"
              >
                Oui
              </b-button>
            </b-col>
          </b-row>
        </b-form>
      </b-modal>

      <!-- MODAL UPDATE USER-->
      <b-modal
        id="modal-update-user"
        title="Modifier l’utilisateur"
        :hide-footer="true"
      >
        <b-form
          @submit.prevent="
            updateUserById(
              $route.params.userId,
              customer.firstname,
              customer.lastname,
              customer.email,
              customer.phonenumber,
              customer.address
            )
          "
        >
```

```
          <b-form-group
            id="input-group-user-firstname"
            label="Prénom :"
            label-for="input-user-firstname"
          >
            <b-form-input
              id="input-user-firstname"
              v-model="customer.firstname"
              type="text"
              placeholder="Entrez le prénom de l’utilisateur"
              required
            ></b-form-input>
          </b-form-group>

          <b-form-group
            id="input-group-user-lastname"
            label="Nom :"
            label-for="input-user-lastname"
          >
            <b-form-input
              id="input-user-lastname"
              v-model="customer.lastname"
              type="text"
              placeholder="Entrez la nom de l’utilisateur"
              required
            ></b-form-input>
          </b-form-group>

          <b-form-group
            id="input-group-user-email"
            label="Adresse e-mail :"
            label-for="input-user-email"
          >
            <b-form-input
              id="input-user-email"
              v-model="customer.email"
              type="email"
              placeholder="Entrez l’adresse e-mail de l’utilisateur"
              required
            ></b-form-input>
          </b-form-group>

          <b-form-group
            id="input-group-user-phonenumber"
            label="Numéro de téléphone :"
            label-for="input-user-phonenumber"
          >
            <b-form-input
              id="input-user-phonenumber"
              v-model="customer.phonenumber"
              type="text"
              placeholder="Entrez le numéro de téléphone de l’utilisateur"
              required
            ></b-form-input>
          </b-form-group>

          <b-form-group
            id="input-group-user-address"
            label="Adresse de domicile :"
            label-for="input-user-address"
          >
            <b-form-input
              id="input-user-address"
              v-model="customer.address"
              type="text"
              placeholder="Entrez l’adresse de domicile de l’utilisateur"
              required
            ></b-form-input>
          </b-form-group>

          <b-button block type="submit" variant="outline-primary">
            Modifier l’utilisateur
          </b-button>
        </b-form>
      </b-modal>

      <!-- MODAL UPDATE DOG-->
      <b-modal
        id="modal-update-dog"
        title="Modifier le chien"
        :hide-footer="true"
      >
        <b-form
          @submit.prevent="
            updateDogById(
              selectedDog.id,
              selectedDog.name,
              selectedDog.breed,
              selectedDog.sex,
              selectedDog.chipId
            )
          "
        >
          <b-form-group
            id="input-group-dog-name"
            label="Nom :"
            label-for="input-dog-name"
          >
            <b-form-input
              id="input-dog-name"
              v-model="selectedDog.name"
              type="text"
              placeholder="Entrez le nom du chien"
              required
            ></b-form-input>
          </b-form-group>

          <b-form-group
            id="input-group-dog-breed"
            label="Race :"
            label-for="input-dog-breed"
          >
            <b-form-input
              id="input-dog-breed"
              v-model="selectedDog.breed"
              type="text"
              placeholder="Entrez la race du chien"
              required
            ></b-form-input>
          </b-form-group>

          <b-form-group
            id="input-group-dog-sex"
            label="Sexe :"
            label-for="input-dog-sex"
          >
            <b-form-select
              id="input-dog-sex"
              v-model="selectedDog.sex"
              :options="sexDogOptions"
              required
            ></b-form-select>
          </b-form-group>

          <b-form-group
            id="input-group-dog-chip-id"
            label="Numéro de puce :"
            label-for="input-dog-chip-id"
          >
            <b-form-input
              id="input-dog-chip-id"
              v-model="selectedDog.chipId"
              type="text"
              placeholder="Entrez le numéro de puce du chien"
              required
            ></b-form-input>
          </b-form-group>

          <b-button block type="submit" variant="outline-primary">
            Modifier le chien
          </b-button>
        </b-form>
      </b-modal>

      <!-- MODAL UPDATE PASSWORD -->
      <b-modal
        id="modal-update-password"
        title="Modifier le mot de passe"
        :hide-footer="true"
      >
        <b-form
          @submit.prevent="updateAuthCustomerPassword(password, repeatPassword)"
        >
          <b-form-group
            id="input-group-inscription-password"
            label="Mot de passe :"
            label-for="input-inscription-password"
          >
            <b-form-input
              id="input-inscription-password"
              v-model="password"
              type="password"
              placeholder="Entrez le mot de passe"
              required
            ></b-form-input>
          </b-form-group>
          <b-form-group
```

```
            id="input-group-inscription-second-password"
            label="Confirmation du mot de passe :"
            label-for="input-inscription-second-password"
          >
            <b-form-input
              id="input-inscription-second-password"
              v-model="repeatPassword"
              type="password"
              placeholder="Entrez le mot de passe"
              required
            ></b-form-input>
          </b-form-group>

          <b-button block type="submit" variant="outline-primary">
            Modifier le mot de passe
          </b-button>
        </b-form>
      </b-modal>
    </b-container>
  </div>
</template>

<script>
import { BIconDownload } from "bootstrap-vue";
import SignaturePad from "./SignaturePad.vue";
import ButtonReturn from "./ButtonReturn.vue";

export default {
  components: {
    BIconDownload,
    SignaturePad,
    ButtonReturn,
  },
  name: "CustomerInformation",
  data() {
    return {
      title: null,
      customer: {
        id: "",
        email: "",
        firstname: "",
        lastname: "",
        phonenumber: "",
        address: "",
        code_role: null,
        dogs: [],
        documents: [],
      },
      addDogForm: {
        name: "",
        breed: "",
        sex: "Mâle",
        chip_id: "",
        pictureFile: null,
      },
      selectedDog: {
        id: null,
        name: "",
        breed: "",
        sex: "",
        chipId: "",
      },
      sexDogOptions: ["Mâle", "Femelle"],
      selectedDocument: {
        id: null,
      },
      addDocumentForm: {
        type: "poster",
        packageNumber: 1,
        file: null,
      },
      documentOptions: [
        { value: "poster", text: "Document pdf" },
        { value: "conditions_inscription", text: "Conditions d’inscription" },
      ],
      packageNumbers: [
        { value: 1, text: "Bilan ’dévaluation : 70 € / 80 CHF" },
        { value: 2, text: "Bilan + 1 séance ’déducation : 125 € / 140 CHF" },
        { value: 3, text: "Forfait bilan + 3 séances : 230 € / 250 CHF" },
        { value: 4, text: "Forfait bilan + 6 séances : 400 € / 440 CHF" },
        { value: 5, text: "Forfait bilan + 9 séances : 520 € / 560 CHF" },
      ],
      signatureBase64: "",
      signatureIsBlocked: false,
      checkboxValidationStatus: false,
      createConditionsButtonisDisabled: true,
      password: null,
      repeatPassword: null,
    };
  },
  methods: {
    /**
     * Method to load all informations of the authenticated client from the api
         rest endpoint "GET api/v1/users/me".
     *
     */
    loadAuthCustomerInformations() {
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token
        },
      };
      this.$http
        .get(this.$API_URL + "users/me/", config)
        .then((response) => {
          this.customer.id = response.data.id;
          this.customer.email = response.data.email;
          this.customer.firstname = response.data.firstname;
          this.customer.lastname = response.data.lastname;
          this.customer.phonenumber = response.data.phonenumber;
          this.customer.address = response.data.address;
          this.customer.code_role = response.data.code_role;
          this.customer.dogs = response.data.dogs;
          this.customer.documents = response.data.documents;
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method to load all informations of a client from his id from the api rest endpoint "GET api/v1/users/{userId}"
     *
     * @param {number} userId The client’s id
     */
    loadCustomerInformationsByUserId(userId) {
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token
        },
      };
      this.$http
        .get(this.$API_URL + "users/" + userId, config)
        .then((response) => {
          console.log(response);
          this.customer.id = response.data.id;
          this.customer.email = response.data.email;
          this.customer.firstname = response.data.firstname;
          this.customer.lastname = response.data.lastname;
          this.customer.phonenumber = response.data.phonenumber;
          this.customer.address = response.data.address;
          this.customer.code_role = response.data.code_role;
          this.customer.dogs = response.data.dogs;
          this.customer.documents = response.data.documents;
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method to create a dog for a user with the api rest endpoint "POST api/v1/dogs".
     *
     * @param {string} name The dog’s name
     * @param {string} breed The dog’s breed
     * @param {string} sex The dog’s sex
     * @param {string} chipId The dog’s chip id
     * @param {number} userId The identifier of the user owner
     */
    createDogForCustomerByUserId(name, breed, sex, chipId, userId) {
      const params = new URLSearchParams();
      params.append("name", name);
      params.append("breed", breed);
      params.append("sex", sex);
      params.append("chip_id", chipId);
      params.append("user_id", userId);
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token,
        },
      };
      this.$http
        .post(this.$API_URL + "dogs/", params, config)
        .then((response) => {
          console.log(response);
          this.loadCustomerInformationsByUserId(this.$route.params.userId);
          this.$alertify.success("Chien ajouté avec succès");
          this.$bvModal.hide("modal-add-dog");
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method to upload a picture dog for a dog from the api rest endpoint "POST api/v1/dogs/uploadPicture".
     *
     * @param {number} id The dog id
     * @param {File} pictureFile The dog picture file
     */
    uploadDogPictureByDogId(id, pictureFile) {
      let formData = new FormData();
      formData.append("dog_picture", pictureFile);
      formData.append("dog_id", id);
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token
        },
      };
      this.$http
        .post(this.$API_URL + "dogs/uploadPicture/", formData, config)
        .then((response) => {
          console.log(response);
          this.loadCustomerInformationsByUserId(this.$route.params.userId);
          this.$alertify.success("Photo du chien ajouté avec succès");
          this.$bvModal.hide("modal-add-picture-dog");
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method to create a document for a user from the api rest endpoint "POST api/v1/documents".
     *
     * @param {string} userId The user id
     * @param {string} type The document type
     * @param {?string} signatureBase64 The signature in base64 format for the conditions of registration
     * @param {?number} packageNumber The package number for the conditions of registration
     * @param {?File} file The PDF document file
     */
    createDocumentForCustomerByUserId(
      userId,
      type,
      signatureBase64,
      packageNumber,
      file
    ) {
      var params;
      if (type == "conditions_inscription") {
        params = new URLSearchParams();
        params.append("signature_base64", signatureBase64);
        params.append("package_number", packageNumber);
      } else {
        params = new FormData();
        params.append("document", file);
      }
      params.append("type", type);
      params.append("user_id", userId);

      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token
        },
      };
      this.$http
        .post(this.$API_URL + "documents/", params, config)
        .then((response) => {
          console.log(response);
          this.loadCustomerInformationsByUserId(this.$route.params.userId);
          this.$alertify.success("Document ajouté avec succès");
          this.$bvModal.hide("modal-add-document");
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method to downlaod a document with the api rest endpoint "GET api/v1/documents/downloadDocument/{documentSerialId}".
     *
     * @param {string} type The document type
     * @param {string} serial_id The document serial id
     */
    downloadDocument(type, serial_id) {
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token,
        },
        responseType: "blob",
      };
      this.$http
        .get(this.$API_URL + "documents/downloadDocument/" + serial_id, config)
        .then((response) => {
          var blob = new Blob([response.data]);
          var file = window.URL.createObjectURL(blob);
          var a = document.createElement("a");
          a.href = file;
          a.download = type + "_" + serial_id + ".pdf";
          document.body.appendChild(a);
          a.click();
          document.body.removeChild(a);
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method to delete a user from his id from the api rest endpoint "DELETE api/v1/users/{userId}"
     *
     * @param {string} id The user id
     */
    deleteUserById(id) {
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token,
        },
      };
      this.$http
        .delete(this.$API_URL + "users/" + id, config)
        .then((response) => {
          this.$alertify.success("Utilisateur supprimé");
          this.$router.push("/administration");
          console.log(response);
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method to delete a dog from his id from the api rest endpoint "DELETE api/v1/dogs/{dogId}"
     *
     * @param {string} id The dog id
     */
    deleteDogById(id) {
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token,
        },
      };
      this.$http
        .delete(this.$API_URL + "dogs/" + id, config)
        .then((response) => {
          this.loadCustomerInformationsByUserId(this.$route.params.userId);
          this.$alertify.success("Chien supprimé");
          console.log(response);
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method to delete a document from his id from the api rest endpoint "DELETE api/v1/documents/{documentId}"
     *
     * @param {string} id The document id
     */
    deleteDocumentById(id) {
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token,
        },
      };
      this.$http
        .delete(this.$API_URL + "documents/" + id, config)
        .then((response) => {
          this.loadCustomerInformationsByUserId(this.$route.params.userId);
          this.$alertify.success("Document supprimé");
          console.log(response);
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method to update a user from his id from the api rest endpoint "PATCH api/v1/users/{userId}"
     *
     * @param {string} id The user id
     * @param {string} firstname The user's first name
     * @param {string} lastname The user's last name
     * @param {string} email The user's email address
     * @param {string} phonenumber The user's phonenumber
     * @param {string} address The user's address
     */
    updateUserById(id, firstname, lastname, email, phonenumber, address) {
      const params = new URLSearchParams();
      params.append("firstname", firstname);
      params.append("lastname", lastname);
      params.append("email", email);
      params.append("phonenumber", phonenumber);
      params.append("address", address);
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token,
        },
      };
      this.$http
        .patch(this.$API_URL + "users/" + id, params, config)
        .then((response) => {
          console.log(response);
          this.loadCustomerInformationsByUserId(this.$route.params.userId);
          this.$alertify.success("Utilisateur modifié");
          this.$bvModal.hide("modal-update-user");
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method to update a dog from his id from the api rest endpoint "PATCH api/v1/dogs/{dogId}"
     *
     * @param {string} id The dog id
     * @param {string} name The dog's name
     * @param {string} breed The dog's breed
     * @param {string} sex The dog's sex
     * @param {string} chipId The dog's chip id
     */
    updateDogById(id, name, breed, sex, chipId) {
      const params = new URLSearchParams();
      params.append("name", name);
      params.append("breed", breed);
      params.append("sex", sex);
      params.append("chip_id", chipId);
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token,
        },
      };
      this.$http
        .patch(this.$API_URL + "dogs/" + id, params, config)
        .then((response) => {
          console.log(response);
          this.loadCustomerInformationsByUserId(this.$route.params.userId);
          this.$alertify.success("Chien modifié");
          this.$bvModal.hide("modal-update-dog");
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method to update the password of the authenticated user with the api rest endpoint "PATCH api/v1/users/me/changePassword".
     *
     * @param {string} password The new password
     * @param {string} repeatPassword Repetition of the new password
     */
    updateAuthCustomerPassword(password, repeatPassword) {
      if (password != repeatPassword) {
        this.$alertify.error("les mots de passes ne corréspondent pas");
        return;
      }
      const params = new URLSearchParams();
      params.append("password", password);
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token,
        },
      };
      this.$http
        .patch(this.$API_URL + "users/me/changePassword/", params, config)
        .then((response) => {
          console.log(response);
          this.$alertify.success("Mot de passe modifié");
          this.$bvModal.hide("modal-update-password");
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    resetModalAddDocument() {
      this.addDocumentForm.type = "poster";
      this.signatureIsBlocked = false;
      this.checkboxValidationStatus = false;
    },
    refreshStateOfcreateConditionsButton() {
      if (this.signatureIsBlocked && this.checkboxValidationStatus) {
        this.createConditionsButtonisDisabled = false;
      } else {
        this.createConditionsButtonisDisabled = true;
      }
    },
    /**
     * Method to allow the component to read the id of a dog in order to proceed to a modification or a deletion from a modal.
     *
     * @param {number} id The dog id
     */
    sendDogId(id) {
      this.selectedDog.id = id;
    },
    /**
     * Method to allow the component to read the data of a dog in order to proceed to a modification or a deletion from a modal.
     *
     * @param {string} name The dog's name
     * @param {string} breed The dog's breed
     * @param {string} sex The dog's sex
     * @param {string} chipId The dog's chip id
     */
    sendDogInformations(name, breed, sex, chipId) {
      this.selectedDog.name = name;
      this.selectedDog.breed = breed;
      this.selectedDog.sex = sex;
      this.selectedDog.chipId = chipId;
    },
    /**
     * Method to allow the component to read the id of a document in order to proceed to a modification or a deletion from a modal.
     *
     * @param {number} id The document id
     */
    sendDocumentId(id) {
      this.selectedDocument.id = id;
    },
    saveSignature(...args) {
      const [base64, isBlocked] = args;
      this.signatureBase64 = base64;
      this.signatureIsBlocked = isBlocked;
    },
  },
  watch: {
    signatureIsBlocked: function () {
      this.refreshStateOfcreateConditionsButton();
    },
  },
  computed: {
    authAdministrator() {
      return this.$store.getters.ifAdministratorAuthenticated;
    },
    authCustomer() {
      return this.$store.getters.ifCustomerAuthenticated;
    },
  },
  mounted() {
    if (this.authAdministrator && this.$route.params.userId) {
      this.title = "Informations du client";
      this.loadCustomerInformationsByUserId(this.$route.params.userId);
    } else {
      this.title = "Mes informations";
      this.loadAuthCustomerInformations();
    }
  },
};
</script>
```

```

<style scoped>
.content {
  margin-top: 20px;
  color: #3ea3d8;
}
.btnAdmin {
  width: 100%;
  margin-bottom: 10px;
}
</style>
```

Listing 9 – ./Sources/src/components/EducatorCalendar.vue

```
<!--
  EducatorCalendar.vue

  Component representing the page of the personal calendar of a dog trainer.

  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
-->

<template>
  <div>
    <b-container>
      <b-row id="title" class="text-center">
        <b-col>
          <h1>Mes rendez-vous</h1>
        </b-col>
      </b-row>
      <b-row>
        <b-col>
          <b-button
            id="btnCreateAppointment"
            style="margin-bottom: 10px"
            variant="outline-primary"
            v-b-modal.modal-create-appointment
            block
          >
            Planifier un rendez-vous
          </b-button>
        </b-col>
      </b-row>
      <b-row class="text-center">
        <b-col>
          <FullCalendar ref="fullCalendar" :options="calendarOptions" />
        </b-col>
      </b-row>

      <!-- MODAL CREATE APPOINTMENT  -->
      <b-modal
        id="modal-create-appointment"
        title="Planifier un rendez-vous"
        :hide-footer="true"
      >
        <b-form
          @submit.prevent="
            createAppointment(
              formCreateAppointment.date,
              formCreateAppointment.timestart,
              formCreateAppointment.durationInHour,
              formCreateAppointment.customer,
              $store.state.user_id
            )
          "
        >
          <b-row>
            <b-col>
              <b-form-group label="Client :">
                <b-form-select
                  id="customerSelect"
                  v-model="formCreateAppointment.customer"
                >
                  <b-form-select-option
                    v-for="customer in customers"
                    :key="customer.id"
                    :value="customer.id"
                    >{{ customer.firstname }}
                    {{ customer.lastname }}</b-form-select-option
                  >
                </b-form-select>
              </b-form-group>
            </b-col>
          </b-row>
          <b-row>
            <b-col class="d-flex justify-content-center">
              <h5>Date</h5>
            </b-col>
          </b-row>
          <b-row>
            <b-col style="margin-bottom: 10px">
              <b-calendar
                block
                id="date-calendar"
                label-help=""
                v-model="formCreateAppointment.date"
              ></b-calendar>
            </b-col>
          </b-row>
          <b-row>
            <b-col>
              <b-form-group
                id="input-group-timeslot-timestart"
                label="L’heure de début :"
                label-for="input-timeslot-timestart"
              >
                <b-form-select
                  id="input-timeslot-timestart"
                  v-model="formCreateAppointment.timestart"
                  :options="timeOptions"
                >
                </b-form-select>
              </b-form-group>
            </b-col>
          </b-row>
          <b-row>
            <b-col>
              <b-form-group
```

```
                id="input-group-timeslot-timeend"
                label="La durée du rendez-vous (en heure) :"
                label-for="input-timeslot-timeend"
              >
                <b-form-select
                  id="input-timeslot-timeend"
                  v-model="formCreateAppointment.durationInHour"
                  :options="timeDurationOptions"
                >
                </b-form-select>
              </b-form-group>
            </b-col>
          </b-row>
          <b-button block type="submit" variant="outline-primary">
            Planifier le rendez-vous
          </b-button>
        </b-form>
      </b-modal>
    </b-container>
  </div>
</template>

<script>
import FullCalendar from "@fullcalendar/vue";
import dayGridPlugin from "@fullcalendar/daygrid";
import timeGridPlugin from "@fullcalendar/timegrid";
import interactionPlugin from "@fullcalendar/interaction";
import frLocale from "@fullcalendar/core/locales/fr";
import moment from "moment";
moment.locale("fr-ch");

export default {
  components: {
    FullCalendar,
  },
  name: "EducatorCalendar",
  data() {
    return {
      calendarOptions: {
        plugins: [dayGridPlugin, timeGridPlugin, interactionPlugin],
        initialView: "dayGridMonth",
        headerToolbar: {
          left: "prev,next today",
          center: "title",
          right: "dayGridMonth,timeGridWeek,timeGridDay",
        },
        height: "auto",
        locale: frLocale,
        eventDisplay: "block",
        allDaySlot: false,
        slotMinTime: "06:00:00",
        slotMaxTime: "20:00:00",
        events: [],
        eventBackgroundColor: "#3ea3d8",
        eventBorderColor: "#e08f25",
        eventClick: (info) => {
          this.$router.push({
            name: "customerAppointment",
            params: { userId: info.event.extendedProps.customerId },
          });
        },
      },
      customers: [],
      formCreateAppointment: {
        customer: null,
        date: null,
        timestart: "07",
        durationInHour: 2,
      },
      timeOptions: [
        { value: "06", text: "06h" },
        { value: "07", text: "07h" },
        { value: "08", text: "08h" },
        { value: "09", text: "09h" },
        { value: "10", text: "10h" },
        { value: "11", text: "11h" },
        { value: "12", text: "12h" },
        { value: "13", text: "13h" },
        { value: "14", text: "14h" },
        { value: "15", text: "15h" },
        { value: "16", text: "16h" },
        { value: "17", text: "17h" },
        { value: "18", text: "18h" },
      ],
      timeDurationOptions: [
        { value: 1, text: "01h" },
        { value: 2, text: "02h" },
        { value: 3, text: "03h" },
        { value: 4, text: "04h" },
      ],
    };
  },
  methods: {
    /**
     * Method to load all the appointments of the authenticated dog educator
     * from the api rest endpoint "GET api/v1/appointments".
     *
     */
    loadEducatorAppointments() {
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token
        },
      };
      this.$http
        .get(this.$API_URL + "appointments/", config)
        .then((response) => {
          this.calendarOptions.events = [];
          response.data.forEach((appointment) => {
            this.calendarOptions.events.push({
              title: "Disponible",
              start: moment(appointment.datetime_appointment).format(
                "YYYY-MM-DD HH:mm:ss"
              ),
              end: moment(appointment.datetime_appointment)
                .add(appointment.duration_in_hour, "hour")
                .format("YYYY-MM-DD HH:mm:ss"),
              extendedProps: {
                customerId: appointment.user_id_customer,
              },
            });
          });
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method to load all customers from the api rest endpoint "GET api/v1/users".
     *
     */
    loadCustomers() {
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token
        },
      };
      this.$http
        .get(this.$API_URL + "users/", config)
        .then((response) => {
          this.customers = response.data;
          this.formCreateAppointment.customer = this.customers[0].id;
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method to create an appointment with the api rest endpoint "POST api/v1/appointments".
     *
     * @param {string} date The date of the appointment
     * @param {string} timestart The time start of the appointment
     * @param {number} durationInHour The duration in hour of the appointment
     * @param {number} user_id_customer The customer's identifier
     * @param {number} user_id_educator The dog educator's identifier
     */
    createAppointment(
      date,
      timestart,
      durationInHour,
      user_id_customer,
      user_id_educator
    ) {
      const params = new URLSearchParams();
      params.append("datetime_appointment", date + " " + timestart + ":00:00");
      params.append("duration_in_hour", durationInHour);
      params.append("user_id_customer", user_id_customer);
      params.append("user_id_educator", user_id_educator);
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token,
        },
      };
      this.$http
        .post(this.$API_URL + "appointments/", params, config)
        .then((response) => {
          console.log(response);
          this.$alertify.success("Rendez-vous reservé avec succès");
          this.$bvModal.hide("modal-create-appointment");
          this.loadEducatorAppointments(this.selected);
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
  },
  mounted() {
    this.loadCustomers();
    this.loadEducatorAppointments();
  },
};
</script>
<style scoped>
#badge-free {
  padding: 10px;
  margin-bottom: 10px;
  background-color: #008000;
  border: #3788d8 1px solid;
}
#title {
  margin-top: 20px;
  color: #3ea3d8;
}
</style>
```

Listing 10 – ./Sources/src/components/Footer.vue

```
<!--
  Footer.vue

  Footer component included in all application previews.

  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
-->

<template>
  <div id="footer">
    <b-container>
      <b-row class="text-center">
        <b-col>
          <h1>Contact</h1>
        </b-col>
      </b-row>
      <b-row class="text-center">
        <b-col cols="12" sm="4">
          <h5>Téléphone</h5>
          <h6><a href="tel:+330624003368">06.24.00.33.68</a></h6>
        </b-col>
        <b-col cols="12" sm="4">
          <h5>Email</h5>
          <h6>
            <a href="mailto:douceurdechien@douceurdechien.com"
              >contact@douceurdechien.com</a
            >
          </h6>
        </b-col>
        <b-col cols="12" sm="4">
          <h5>Zone d'intervention</h5>
          <h6>Montréjeau et alentours</h6>
        </b-col>
      </b-row>
    </b-container>
    <b-container fluid>
      <b-row class="text-center">
        <b-col style="background-color: white; color: #3ea3d8">©
          2021 Douceur de chien -
          <RouterLink class="link" to="/privacy_policy"
            >politique de confidentialité</RouterLink
          >
          -
          <RouterLink class="link" to="/privacy_policy"
            >conditions d'utilisation
          </RouterLink>
        </b-col>
      </b-row>
    </b-container>
  </div>
</template>

<script>
export default {
  name: "Footer",
};
</script>

<style scoped>
a {
  color: white;
  text-decoration: underline;
}
#footer {
  background-color: #3ea3d8;
  color: white;
  margin-top: 10px;
}
.link {
  color: #0051ff;
}
</style>
```

Listing 11 – ./Sources/src/components/Home.vue

```
<!--
  Home.vue

  Component representing the home page of the application.

  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
-->

<template>
  <div>
    <left-section-content
      :textH1="texth1LeftSectionContent"
      :textH3="texth3LeftSectionContent"
      :textH5="texth5LeftSectionContent"
      :textH6="texth6LeftSectionContent"
      :routeBouton="routeBoutonLeftSectionContent"
      :textButton="textButtonLeftSectionContent"
      :imageSource="imageSourceLeftSectionContent"
    ></left-section-content>
    <right-section-content
      :textH3="texth3RightSectionContent"
      :textParagraph="textParagraphRightSectionContent"
      :routeBouton="routeBoutonRightSectionContent"
      :textButton="textButtonRightSectionContent"
      :imageSource="imageSourceRightSectionContent"
    ></right-section-content>
  </div>
</template>

<script>
import LeftSectionContent from "./LeftSectionContent.vue";
import Hyron from "../assets/img/hyron.png";
import RightSectionContent from "./RightSectionContent.vue";
import BorisWithDogs from "../assets/img/borisWithDogs.png";

export default {
  components: { LeftSectionContent, RightSectionContent },
  name: "Home",
  data() {
    return {
      texth1LeftSectionContent: "Douceur de chien",
      texth3LeftSectionContent: "Educateur & comportementaliste canin",
      texth5LeftSectionContent: "Vous voulez prendre rendez-vous ?",
      texth6LeftSectionContent: "Accéder aux agendas des éducateurs canins",
      textButtonLeftSectionContent: "Agendas",
      imageSourceLeftSectionContent: Hyron,
      routeBoutonLeftSectionContent: "calendar",

      texth3RightSectionContent: "À propos de nous",
      textButtonRightSectionContent: "Lire plus",
      textParagraphRightSectionContent:
        "Je m'appelle Boris Gourdoux, je suis le fondateur de Douceur de Chien. Accompagné de Hyron, Staffie mâle et Jaya, Rodhesian Ridgeback femelle. J'ai fais de l'éducation bienveillante et moderne mon leitmotiv afin d'aider tous les maîtres de chiens qui feront appel à mes services. Formé et diplômé en rééducation comportementale, je me ferais un plaisir de vous accompagner dans l'éducation de votre chien.",
      imageSourceRightSectionContent: BorisWithDogs,
      routeBoutonRightSectionContent: "about",
    };
  },
};
</script>

<style scoped></style>
```

Listing 12 – ./Sources/src/components/Inscription.vue

```
<!--
  Inscription.vue

  Component representing the inscription page of the application.

  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
-->

<template>
  <div class="content">
    <b-container>
      <b-row id="title" class="text-center">
        <b-col>
          <h1>S’inscrire</h1>
        </b-col>
      </b-row>
      <b-row class="text-center text-secondary">
        <b-col>
          Déjà membre ?
          <router-link to="/connection">
            <span role="link">Se connecter</span>
          </router-link>
        </b-col>
      </b-row>
      <b-row id="title" class="justify-content-center">
        <b-col lg="6">
          <b-form @submit.prevent="inscription">
            <b-form-group
              id="input-group-inscription-email"
              label="Adresse e-mail :"
              label-for="input-inscription-email"
            >
              <b-form-input
                id="input-inscription-email"
                v-model="form.email"
                type="email"
                placeholder="Entrez l’adresse e-mail"
                required
              ></b-form-input>
            </b-form-group>
            <b-form-group
              id="input-group-inscription-firstname"
              label="Prénom :"
              label-for="input-inscription-firstname"
            >
              <b-form-input
                id="input-inscription-firstname"
                v-model="form.firstname"
                type="text"
                placeholder="Entrez votre prénom"
                required
              ></b-form-input>
            </b-form-group>
            <b-form-group
              id="input-group-inscription-lastname"
              label="Nom de famille :"
              label-for="input-inscription-lastname"
            >
              <b-form-input
                id="input-inscription-lastname"
                v-model="form.lastname"
                type="text"
                placeholder="Entrez votre nom de famille"
                required
              ></b-form-input>
            </b-form-group>
            <b-form-group
              id="input-group-inscription-phonenumber"
              label="Numéro de téléphone :"
              label-for="input-inscription-phonenumber"
            >
              <b-form-input
                id="input-inscription-phonenumber"
                v-model="form.phonenumber"
                type="text"
                placeholder="Entrez votre numéro de téléphone"
                required
              ></b-form-input>
            </b-form-group>
            <b-form-group
              id="input-group-inscription-address"
              label="Adresse :"
              label-for="input-inscription-address"
            >
              <b-form-input
                id="input-inscription-address"
                v-model="form.address"
                type="text"
                placeholder="Entrez votre addresse"
                required
              ></b-form-input>
            </b-form-group>
            <b-form-group
              id="input-group-inscription-password"
              label="Mot de passe :"
              label-for="input-inscription-password"
            >
              <b-form-input
                id="input-inscription-password"
                v-model="form.password"
                type="password"
                placeholder="Entrez le mot de passe"
                required
              ></b-form-input>
            </b-form-group>
            <b-form-group
              id="input-group-inscription-second-password"
              label="Confirmation du mot de passe :"
              label-for="input-inscription-second-password"
            >
              <b-form-input
                id="input-inscription-second-password"
                v-model="form.repeatPassword"
                type="password"
                placeholder="Entrez le mot de passe"
                required
              ></b-form-input>
            </b-form-group>
            <b-form-group
              id="input-group-inscription-captcha"
              label="Captcha :"
              label-for="input-inscription-captcha"
            >
              <label for="robot"></label>
              <vue-recaptcha
                ref="recaptcha"
                @verify="onVerify"
                sitekey="6LfeMhEbAAAAAHuNfrr-lGCFGw8yYjkYAEb47y0w"
                :loadRecaptchaScript="true"
              >
              </vue-recaptcha>
            </b-form-group>
            <b-button block type="submit" variant="outline-primary">
              Inscription
            </b-button>
          </b-form>
        </b-col>
      </b-row>
    </b-container>
  </div>
</template>

<script>
import VueRecaptcha from "vue-recaptcha";

export default {
  name: "Inscription",
  components: {
    VueRecaptcha,
  },
  data() {
    return {
      form: {
        email: "",
        firstname: "",
        lastname: "",
        phonenumber: "",
        address: "",
        password: "",
        repeatPassword: "",
        robot: false,
        reCAPTCHAresponse: "",
      },
    };
  },
  methods: {
    onVerify(response) {
      if (response) {
        this.form.robot = true;
        this.form.reCAPTCHAresponse = response;
      }
    },
    /**
     * Method to register a user with the store.
     *
     */
    inscription() {
      if (this.form.password != this.form.repeatPassword) {
        this.$alertify.error("les mots de passes ne corréspondent pas");
        return;
      }

      if (!this.form.robot) {
        this.$alertify.error("la vérification du captcha a échoué");
        return;
      }

      this.$store.dispatch("inscription", {
        email: this.form.email,
        firstname: this.form.firstname,
        lastname: this.form.lastname,
        phonenumber: this.form.phonenumber,
        address: this.form.address,
        password: this.form.password,
        reCAPTCHAuserResponseToken: this.form.reCAPTCHAresponse,
      });
    },
  },
};
</script>

<style scoped>
#title {
  margin-top: 20px;
  color: #3ea3d8;
}
</style>
```

Listing 13 – ./Sources/src/components/LeftSectionContent.vue

```
<!--
  LeftSectionContent.vue

  Reusable component representing an aesthetic content displayed with textual content on the left and an image on the right.

  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
-->

<template>
  <div class="content">
    <b-container>
      <b-row>
        <b-col class="text-content" cols="12" sm="12" lg="6">
          <h1 class="font-weight-bold display-4">{{ textH1 }}</h1>
          <h3 class="font-weight-bold">{{ textH3 }}</h3>
          <h5>{{ textH5 }}</h5>
          <h6>{{ textH6 }}</h6>
          <p class="text-justify">{{ textParagraph }}</p>
          <b-button id="button" :to="routeBouton" pill v-if="textButton != ’’">
            {{ textButton }}</b-button
          >
        </b-col>
        <b-col
          class="d-flex align-items-center justify-content-center"
          cols="12"
          sm="12"
          lg="6"
        >
```

```
          <b-img
            v-if="imageSource != null"
            class="circleImage"
            :src="imageSource"
            fluid
            rounded="circle"
            alt="Circle image"
          ></b-img>
        </b-col>
      </b-row>
    </b-container>
  </div>
</template>

<script>
export default {
  props: {
    textParagraph: {
      type: String,
      default: "",
    },
    textH1: {
      type: String,
      default: "",
    },
    textH3: {
      type: String,
      default: "",
    },
    textH5: {
      type: String,
      default: "",
    },
    textH6: {
      type: String,
      default: "",
    },
    routeBouton: {
      type: String,
      default: "",
    },
    textButton: {
      type: String,
      default: "",
    },
    imageSource: {
      type: String,
      default: "",
    },
  },
  name: "LeftSectionContent",
};
</script>

<style scoped>
.content {
  background-color: #fef6ec;
  padding-top: 20px;
  padding-bottom: 20px;
  color: #3ea3d8;
}
#button {
  background-color: #3ea3d8;
}
</style>
```

Listing 14 – ./Sources/src/components/Navbar.vue

```
<!--
  Navbar.vue

  Navbar component included in all application previews.

  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
-->

<template>
  <div>
    <b-navbar fixed="top" toggleable="lg" type="dark" variant="dark">
      <b-img
        src="../assets/img/logoblancblanc.png"
        v-bind="logo"
        fluid
        @click="$router.push('/')"
        alt="Responsive image"
      ></b-img>
      <b-navbar-toggle target="nav-collapse"></b-navbar-toggle>
      <b-collapse id="nav-collapse" is-nav>
        <b-navbar-nav>
          <b-nav-item to="/" href="#">Accueil</b-nav-item>
          <b-nav-item to="/about" href="#">À propos</b-nav-item>
          <b-nav-item to="/calendar" href="#">Agenda</b-nav-item>
          <b-nav-item to="/customer_information" v-if="authCustomer" href="#">
            Mes informations
          </b-nav-item>
          <b-nav-item to="/customer_appointment" v-if="authCustomer" href="#">
            Mes rendez-vous
          </b-nav-item>
          <b-nav-item to="/administration" v-if="authAdministrator" href="#">
            Administration
          </b-nav-item>
          <b-nav-item to="/planning" v-if="authAdministrator" href="#">
            Mon planning
          </b-nav-item>
          <b-nav-item to="/educator_calendar" v-if="authAdministrator" href="#">
            Mes rendez-vous
          </b-nav-item>
        </b-navbar-nav>
        <b-navbar-nav class="ml-auto">
          <b-nav-item v-if="auth" @click="logout()" href="#">
            Déconnexion
          </b-nav-item>
          <b-nav-item v-if="!auth" to="/connection" href="#">
            Connexion
          </b-nav-item>
        </b-navbar-nav>
      </b-collapse>
    </b-navbar>
  </div>
</template>

<script>
export default {
  name: "Navbar",
  computed: {
    auth() {
      return this.$store.getters.ifAuthenticated;
    },
    authCustomer() {
      return this.$store.getters.ifCustomerAuthenticated;
    },
    authAdministrator() {
      return this.$store.getters.ifAdministratorAuthenticated;
    },
  },
  methods: {
    logout() {
      this.$store.dispatch("logout");
    },
  },
  data() {
    return {
      logo: { width: 80 },
    };
  },
};
</script>

<style scoped>
.navbar.navbar-dark.bg-dark {
  background-color: #3ea3d8 !important;
  font-size: 20px;
}
.navbar-dark .navbar-nav .nav-link {
  color: white;
  -webkit-transition: color 0.2s ease-in;
  -moz-transition: color 0.2s ease-in;
  -o-transition: color 0.2s ease-in;
  transition: color 0.2s ease-in;
  text-align: center;
}
.navbar-dark .navbar-nav .router-link-exact-active {
  color: #e08f25;
  font-weight: 500;
}
.navbar-dark .navbar-nav .nav-link:hover {
  color: #e08f25;
  font-weight: 500;
}
.navbar-collapse .nav-item .nav-link {
  margin: auto;
  font-weight: 500;
}
</style>
```

Listing 15 – ./Sources/src/components/Planning.vue

```
<!--
  Planning.vue

  Component representing the planning page of a dog trainer.

  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
-->

<template>
  <div>
    <b-container>
      <b-row>
        <b-col>
          <h1 class="title text-center">Mon planning</h1>
        </b-col>
      </b-row>
      <b-row>
        <b-col>
          <h3 class="title text-center">Calendriers hebdomadaires</h3>
        </b-col>
      </b-row>
      <b-row>
        <b-table
          table-variant="primary"
          striped
          hover
          :items="weeklySchedules"
          :fields="tblWeeklySchedulesFields"
        >
          <template #cell(créneaux_horaires)="row">
            <b-button
              size="sm"
              @click="row.toggleDetails"
              class="mr-2 tableBtn"
            >
              {{ row.detailsShowing ? "Cacher" : "Afficher" }}
            </b-button>
          </template>
          <template #row-details="row">
            <b-table
              table-variant="light"
              striped
              hover
              :items="row.item.timeSlots"
              :fields="tblTimeSlotsWeeklySchedulesFields"
            >
              <template #cell(supprimer)="row">
                <b-button
                  v-b-modal.modal-delete-timeslot
                  @click="sendTimeSlotId(row.item.id)"
                  variant="danger"
                >
                  <b-icon-trash></b-icon-trash>
                </b-button>
              </template>
            </b-table>
            <b-button
              variant="primary"
              class="bg-light text-primary"
              v-b-modal.modal-add-timeslot-weeklyschedule
              block
              @click="sendWeeklyScheduleId(row.item.id)"
            >
              Ajouter un créneau horaire
            </b-button>
          </template>
          <template #cell(supprimer)="row">
            <b-button
              v-b-modal.modal-delete-weeklyschedule
              @click="sendWeeklyScheduleId(row.item.id)"
              variant="danger"
            >
              <b-icon-trash></b-icon-trash>
            </b-button>
          </template>
        </b-table>
      </b-row>
      <b-row>
        <b-col>
          <b-button
            id="btnAddWeeklySchedule"
            v-b-modal.modal-add-weeklyschedule
            variant="primary"
            block
          >
            Ajouter un calendrier hebdomadaire
          </b-button>
        </b-col>
      </b-row>
      <b-row>
        <b-col>
          <h3 class="title text-center">Exceptions d'horaire</h3>
        </b-col>
      </b-row>
      <b-row>
        <b-table
          table-variant="primary"
          striped
          hover
          :items="scheduleOverrides"
          :fields="tblScheduleOverridesFields"
        >
          <template #cell(créneaux_horaires)="row">
            <b-button
              size="sm"
              @click="row.toggleDetails"
              class="mr-2 tableBtn"
            >
              {{ row.detailsShowing ? "Cacher" : "Afficher" }}
            </b-button>
          </template>
          <template #row-details="row">
            <b-table
              table-variant="light"
              striped
              hover
              :items="row.item.timeSlots"
              :fields="tblTimeSlotsScheduleOverridesFields"
            >
              <template #cell(supprimer)="row">
                <b-button
                  v-b-modal.modal-delete-timeslot
                  @click="sendTimeSlotId(row.item.id)"
                  variant="danger"
                >
                  <b-icon-trash></b-icon-trash>
                </b-button>
              </template>
            </b-table>
            <b-button
              variant="primary"
              class="bg-light text-primary"
              v-b-modal.modal-add-timeslot-scheduleoverride
              block
              @click="sendScheduleOverrideId(row.item.id)"
            >
              Ajouter un créneau horaire
            </b-button>
          </template>
          <template #cell(supprimer)="row">
            <b-button
              v-b-modal.modal-delete-scheduleoverride
              @click="sendScheduleOverrideId(row.item.id)"
              variant="danger"
            >
              <b-icon-trash></b-icon-trash>
            </b-button>
          </template>
        </b-table>
      </b-row>
      <b-row>
        <b-col>
          <b-button
            id="btnAddScheduleOverride"
            v-b-modal.modal-add-scheduleoverride
            variant="primary"
            block
          >
            Ajouter une exception d'horaire
          </b-button>
        </b-col>
      </b-row>

      <b-row>
        <b-col>
          <h3 class="title text-center">Vacances</h3>
        </b-col>
      </b-row>
      <b-row>
        <b-table
          table-variant="primary"
          striped
          hover
          :items="absences"
          :fields="tblAbsencesFields"
        >
          <template #cell(supprimer)="row">
            <b-button
              v-b-modal.modal-delete-absence
              @click="sendAbsenceId(row.item.id)"
              variant="danger"
            >
              <b-icon-trash></b-icon-trash>
            </b-button>
          </template>
        </b-table>
      </b-row>
      <b-row>
        <b-col>
          <b-button
            id="btnAddAbsence"
            v-b-modal.modal-add-absence
            variant="primary"
            block
          >
            Ajouter des vacances
          </b-button>
        </b-col>
      </b-row>

      <!-- MODAL ADD WEEKLYSCHEDULE -->
      <b-modal
        id="modal-add-weeklyschedule"
        title="Ajouter un calendrier hebdomadaire"
        :hide-footer="true"
      >
        <b-form
          @submit.prevent="
            createWeeklySchedule(
              formWeeklySchedule.date_valid_from,
              formWeeklySchedule.date_valid_to
            )
          "
        >
          <b-form-group
            id="input-group-weeklyschedule-datevalidfrom"
            label="Date de début :"
            label-for="input-weeklyschedule-datevalidfrom"
          >
            <b-form-datepicker
              id="input-weeklyschedule-datevalidfrom"
              v-model="formWeeklySchedule.date_valid_from"
              menu-class="w-100"
              label-no-date-selected="Aucune date selectionnée"
              calendar-width="100%"
              label-help=""
              class="mb-2"
            >
            </b-form-datepicker>
          </b-form-group>
          <b-form-group
            id="input-group-weeklyschedule-datevalidto"
            label="Date de fin :"
            label-for="input-weeklyschedule-datevalidto"
          >
            <b-form-datepicker
```

```
              id="input-weeklyschedule-datevalidto"
              v-model="formWeeklySchedule.date_valid_to"
              menu-class="w-100"
              label-no-date-selected="Aucune date selectionnée"
              calendar-width="100%"
              label-help=""
              class="mb-2"
            >
            </b-form-datepicker>
            <b-button
              size="sm"
              variant="outline-danger"
              v-if="formWeeklySchedule.date_valid_to"
              @click="clearWeeklyScheduleDateValidTo"
            >
              Effacer la date
            </b-button>
          </b-form-group>

          <b-button block type="submit" variant="outline-primary">
            Ajouter le calendrier hebdomadaire
          </b-button>
        </b-form>
      </b-modal>

      <!-- MODAL ADD SCHEDULEOVERRIDE -->
      <b-modal
        id="modal-add-scheduleoverride"
        title="Ajouter un exception d'horaire"
        :hide-footer="true"
      >
        <b-form
          @submit.prevent="
            createScheduleOverride(formScheduleOverride.date_schedule_override)
          "
        >
          <b-form-group
            id="input-group-scheduleoverride-datescheduleoverride"
            label="Date de début :"
            label-for="input-scheduleoverride-datescheduleoverride"
          >
            <b-form-datepicker
              id="input-scheduleoverride-datescheduleoverride"
              v-model="formScheduleOverride.date_schedule_override"
              menu-class="w-100"
              label-no-date-selected="Aucune date selectionnée"
              calendar-width="100%"
              label-help=""
              class="mb-2"
            >
            </b-form-datepicker>
          </b-form-group>

          <b-button block type="submit" variant="outline-primary">
            Ajouter l'exception d'horaire
          </b-button>
        </b-form>
      </b-modal>

      <!-- MODAL ADD ABSENCE -->
      <b-modal
        id="modal-add-absence"
        title="Ajouter des vacances"
        :hide-footer="true"
      >
        <b-form
          @submit.prevent="
            createAbsence(
              formAbsence.date_absence_from,
              formAbsence.date_absence_to
            )
          "
        >
          <b-form-group
            id="input-group-absence-dateabsencefrom"
            label="Date de début :"
            label-for="input-absence-dateabsencefrom"
          >
            <b-form-datepicker
              id="input-absence-dateabsencefrom"
              v-model="formAbsence.date_absence_from"
              menu-class="w-100"
              label-no-date-selected="Aucune date selectionnée"
              calendar-width="100%"
              label-help=""
              class="mb-2"
            >
            </b-form-datepicker>
          </b-form-group>

          <b-form-group
            id="input-group-absence-dateabsenceto"
            label="Date de début :"
            label-for="input-absence-dateabsenceto"
          >
            <b-form-datepicker
              id="input-absence-dateabsenceto"
              v-model="formAbsence.date_absence_to"
              menu-class="w-100"
              label-no-date-selected="Aucune date selectionnée"
              calendar-width="100%"
              class="mb-2"
            >
            </b-form-datepicker>
          </b-form-group>

          <b-button block type="submit" variant="outline-primary">
            Ajouter les vacances
          </b-button>
        </b-form>
      </b-modal>

      <!-- MODAL ADD TIMESLOT FOR WEEKLYSCHEDULE -->
      <b-modal
        id="modal-add-timeslot-weeklyschedule"
        title="Ajouter un créneau horaire"
        :hide-footer="true"
      >
```

```
        <b-form
          @submit.prevent="
            createTimeSlotForWeeklySchedule(
              selectedWeeklyScheduleId,
              formTimeSlot.code_day,
              formTimeSlot.time_start,
              formTimeSlot.time_end
            )
          "
        >
          <b-form-group
            id="input-group-timeslot-codeday"
            label="Jour de la semaine :"
            label-for="input-timeslot-codeday"
          >
            <b-form-select
              id="input-timeslot-codeday"
              v-model="formTimeSlot.code_day"
              :options="dayOptions"
            >
            </b-form-select>
          </b-form-group>
          <b-form-group
            id="input-group-timeslot-timestart"
            label="L'heure de début :"
            label-for="input-timeslot-timestart"
          >
            <b-form-select
              id="input-timeslot-timestart"
              v-model="formTimeSlot.time_start"
              :options="timeOptions"
            >
            </b-form-select>
          </b-form-group>
          <b-form-group
            id="input-group-timeslot-timeend"
            label="L'heure de fin :"
            label-for="input-timeslot-timeend"
          >
            <b-form-select
              id="input-timeslot-timeend"
              v-model="formTimeSlot.time_end"
              :options="timeOptions"
            >
            </b-form-select>
          </b-form-group>

          <b-button block type="submit" variant="outline-primary">
            Ajouter le créneau horaire
          </b-button>
        </b-form>
      </b-modal>

      <!-- MODAL ADD TIMESLOT FOR SCHEDULE OVERRIDE -->
      <b-modal
        id="modal-add-timeslot-scheduleoverride"
        title="Ajouter un créneau horaire"
        :hide-footer="true"
      >
```

```
        <b-form
          @submit.prevent="
            createTimeSlotForScheduleOverride(
              selectedScheduleOverrideId,
              formTimeSlot.time_start,
              formTimeSlot.time_end
            )
          "
        >
          <b-form-group
            id="input-group-timeslot-timestart"
            label="L’heure de début :"
            label-for="input-timeslot-timestart"
          >
            <b-form-select
              id="input-timeslot-timestart"
              v-model="formTimeSlot.time_start"
              :options="timeOptions"
            >
            </b-form-select>
          </b-form-group>
          <b-form-group
            id="input-group-timeslot-timeend"
            label="L’heure de fin :"
            label-for="input-timeslot-timeend"
          >
            <b-form-select
              id="input-timeslot-timeend"
              v-model="formTimeSlot.time_end"
              :options="timeOptions"
            >
            </b-form-select>
          </b-form-group>

          <b-button block type="submit" variant="outline-primary">
            Ajouter le créneau horaire
          </b-button>
        </b-form>
      </b-modal>

      <!-- MODAL DELETE WEEKLY SCHEDULE-->
      <b-modal
        id="modal-delete-weeklyschedule"
        title="Confirmation de supression d’un calendrier hebdomadaire"
        :hide-footer="true"
        :hide-header="true"
      >
        <h5 style="text-align: center">
          Supprimer le calendrier hebdomadaire ?
        </h5>
        <b-form
          @submit.prevent="deleteWeeklyScheduleById(selectedWeeklyScheduleId)"
        >
          <b-row>
            <b-col>
              <b-button
                block
                variant="danger"
                @click="$bvModal.hide(’modal-delete-weeklyschedule’)"
              >
                Non
              </b-button>
            </b-col>
            <b-col>
              <b-button
                @click="$bvModal.hide(’modal-delete-weeklyschedule’)"
                block
                type="submit"
                variant="success"
              >
                Oui
              </b-button>
            </b-col>
          </b-row>
        </b-form>
      </b-modal>

      <!-- MODAL DELETE SCHEDULE OVERRIDE-->
      <b-modal
        id="modal-delete-scheduleoverride"
        title="Confirmation de supression d’une exception d’horaire"
        :hide-footer="true"
        :hide-header="true"
      >
        <h5 style="text-align: center">Supprimer l’exception d’horaire ?</h5>
        <b-form
          @submit.prevent="
            deleteScheduleOverrideById(selectedScheduleOverrideId)
          "
        >
          <b-row>
            <b-col>
              <b-button
                block
                variant="danger"
                @click="$bvModal.hide(’modal-delete-scheduleoverride’)"
              >
                Non
              </b-button>
            </b-col>
            <b-col>
              <b-button
                @click="$bvModal.hide(’modal-delete-scheduleoverride’)"
                block
                type="submit"
                variant="success"
              >
                Oui
              </b-button>
            </b-col>
          </b-row>
        </b-form>
      </b-modal>

      <!-- MODAL DELETE ABSENCE-->
      <b-modal
        id="modal-delete-absence"
        title="Confirmation de supression des vacances"
        :hide-footer="true"
        :hide-header="true"
      >
        <h5 style="text-align: center">Supprimer les vacances ?</h5>
        <b-form @submit.prevent="deleteAbsenceById(selectedAbsenceId)">
          <b-row>
            <b-col>
              <b-button
                block
                variant="danger"
                @click="$bvModal.hide(’modal-delete-absence’)"
              >
                Non
              </b-button>
            </b-col>
            <b-col>
              <b-button
                @click="$bvModal.hide(’modal-delete-absence’)"
                block
                type="submit"
                variant="success"
              >
                Oui
              </b-button>
            </b-col>
          </b-row>
        </b-form>
      </b-modal>

      <!-- MODAL DELETE TIME SLOT -->
      <b-modal
        id="modal-delete-timeslot"
        title="Confirmation de supression d’un créneau horaire"
        :hide-footer="true"
        :hide-header="true"
      >
        <h5 style="text-align: center">Supprimer le créneau horaire ?</h5>
        <b-form @submit.prevent="deleteTimeSlotById(selectedTimeSlotId)">
          <b-row>
            <b-col>
              <b-button
                block
                variant="danger"
                @click="$bvModal.hide(’modal-delete-timeslot’)"
              >
                Non
              </b-button>
            </b-col>
            <b-col>
              <b-button
                @click="$bvModal.hide(’modal-delete-timeslot’)"
                block
                type="submit"
                variant="success"
              >
                Oui
              </b-button>
            </b-col>
          </b-row>
        </b-form>
      </b-modal>
    </b-container>
  </div>
</template>

<script>
import { BIconTrash } from "bootstrap-vue";
import moment from "moment";
moment.locale("fr-ch");

export default {
  components: {
    BIconTrash,
  },
  name: "Planning",
  data() {
    return {
      weeklySchedules: [],
      timeSlots: [],
      scheduleOverrides: [],
      absences: [],
      selectedWeeklyScheduleId: null,
      selectedScheduleOverrideId: null,
      selectedTimeSlotId: null,
      selectedAbsenceId: null,
      formTimeSlot: {
        code_day: 2,
        time_start: "07",
        time_end: "09",
      },
      formWeeklySchedule: {
        date_valid_from: null,
        date_valid_to: null,
      },
      formScheduleOverride: {
        date_schedule_override: null,
      },
      formAbsence: {
        date_absence_from: null,
        date_absence_to: null,
      },
      dayOptions: [
        { value: 2, text: "Lundi" },
        { value: 3, text: "Mardi" },
        { value: 4, text: "Mercredi" },
        { value: 5, text: "Jeudi" },
        { value: 6, text: "Vendredi" },
        { value: 7, text: "Samedi" },
        { value: 1, text: "Dimanche" },
      ],
      timeOptions: [
        { value: "06", text: "06h" },
        { value: "07", text: "07h" },
        { value: "08", text: "08h" },
        { value: "09", text: "09h" },
        { value: "10", text: "10h" },
        { value: "11", text: "11h" },
        { value: "12", text: "12h" },
        { value: "13", text: "13h" },
        { value: "14", text: "14h" },
        { value: "15", text: "15h" },
        { value: "16", text: "16h" },
        { value: "17", text: "17h" },
        { value: "18", text: "18h" },
      ],
      dayOfWeek: [
        "",
        "Dimanche",
        "Lundi",
        "Mardi",
        "Mercredi",
        "Jeudi",
        "Vendredi",
        "Samedi",
      ],
      tblWeeklySchedulesFields: [
        "date_de_début",
        "date_de_fin",
        "créneaux_horaires",
        "supprimer",
      ],
      tblScheduleOverridesFields: ["date", "créneaux_horaires", "supprimer"],
      tblAbsencesFields: ["date_de_début", "date_de_fin", "supprimer"],
      tblTimeSlotsWeeklySchedulesFields: [
        "jour",
        "heure_de_début",
        "heure_de_fin",
        "supprimer",
      ],
      tblTimeSlotsScheduleOverridesFields: [
        "heure_de_début",
        "heure_de_fin",
        "supprimer",
      ],
    };
  },
  methods: {
    /**
     * Method to load and format all the weekly schedules with their time
     * slots of the authenticated dog educator from the api rest endpoint "GET api/v1/weeklySchedules".
     *
     */
    loadEducatorWeeklySchedulesWithTimeSlots() {
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token,
        },
      };
      this.$http
        .get(this.$API_URL + "weeklySchedules/", config)
        .then((response) => {
          this.weeklySchedules = [];
          response.data.forEach((weeklySchedule) => {
            weeklySchedule["date_de_début"] = this.formatDate(
              weeklySchedule["date_valid_from"]
            );
            delete weeklySchedule["date_valid_from"];
            weeklySchedule["date_de_fin"] =
              weeklySchedule["date_valid_to"] == undefined
                ? "-"
                : this.formatDate(weeklySchedule["date_valid_to"]);
            delete weeklySchedule["date_valid_to"];

            weeklySchedule["timeSlots"].forEach((timeslot) => {
              timeslot["jour"] = this.dayOfWeek[timeslot["code_day"]];
              delete timeslot["code_day"];
              timeslot["heure_de_début"] = this.formatTime(
                timeslot["time_start"]
              );
              delete timeslot["time_start"];
              timeslot["heure_de_fin"] = this.formatTime(timeslot["time_end"]);
              delete timeslot["time_end"];
            });
          });
          this.weeklySchedules = response.data;
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method to load and format all the schedule overrides with their time slots
     * of the authenticated dog educator from the api rest endpoint "GET api/v1/scheduleOverrides".
     *
     */
    loadEducatorScheduleOverridesWithTimeSlots() {
      const config = {
        headers: {
          // eslint-disable-next-line prettier/prettier
          "Authorization" : this.$store.state.api_token,
        },
      };
      this.$http
        .get(this.$API_URL + "scheduleOverrides/", config)
        .then((response) => {
          this.scheduleOverrides = [];
          response.data.forEach((scheduleOverride) => {
            scheduleOverride["date"] = this.formatDate(
              scheduleOverride["date_schedule_override"]
            );
            delete scheduleOverride["date_schedule_override"];

            scheduleOverride["timeSlots"].forEach((timeslot) => {
              timeslot["heure_de_début"] = this.formatTime(
                timeslot["time_start"]
              );
              delete timeslot["time_start"];
              timeslot["heure_de_fin"] = this.formatTime(timeslot["time_end"]);
              delete timeslot["time_end"];
            });
          });
          this.scheduleOverrides = response.data;
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method to load and format all absences of the authenticated dog educator from the api rest endpoint "GET api/v1/scheduleOverrides".
     *
     */
    loadEducatorAbsences() {
      const config = {
        headers: {
          // eslint-disable-next-line prettier/prettier
          "Authorization" : this.$store.state.api_token,
        },
      };
      this.$http
        .get(this.$API_URL + "absences/", config)
        .then((response) => {
          this.absences = [];
          response.data.forEach((absence) => {
            absence["date_de_début"] = this.formatDate(
              absence["date_absence_from"]
            );
            delete absence["date_absence_from"];
            absence["date_de_fin"] = this.formatDate(
              absence["date_absence_to"]
            );
            delete absence["date_absence_to"];
          });
          this.absences = response.data;
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method to create a time slot for a weekly schedule from the api rest endpoint "POST api/v1/timeSlots".
     *
     * @param {string} weeklyScheduleId The weekly schedule id
     * @param {string} codeDay The code day of time slot
     * @param {string} timeStart The time start of time slot
     * @param {?number} timeEnd The time end of time slot
     */
    createTimeSlotForWeeklySchedule(
      weeklyScheduleId,
      codeDay,
      timeStart,
      timeEnd
    ) {
      const params = new URLSearchParams();
      params.append("id_weekly_schedule", weeklyScheduleId);
      params.append("code_day", codeDay);
      params.append("time_start", timeStart + ":00:00");
      params.append("time_end", timeEnd + ":00:00");
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token,
        },
      };
      this.$http
        .post(this.$API_URL + "timeSlots/", params, config)
        .then((response) => {
          console.log(response);
          this.loadEducatorWeeklySchedulesWithTimeSlots();
          this.$alertify.success("Créneau horaire ajouté avec succès");
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method to create a time slot for a schedule override from the api rest endpoint "POST api/v1/timeSlots".
     *
     * @param {string} weeklyScheduleId The weekly schedule id
     * @param {string} codeDay The code day of time slot
     * @param {string} timeStart The time start of time slot
     * @param {?number} timeEnd The time end of time slot
     */
    createTimeSlotForScheduleOverride(scheduleOverrideId, timeStart, timeEnd) {
      const params = new URLSearchParams();
      params.append("id_schedule_override", scheduleOverrideId);
      params.append("code_day", 1);
      params.append("time_start", timeStart + ":00:00");
      params.append("time_end", timeEnd + ":00:00");
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token,
        },
      };
      this.$http
        .post(this.$API_URL + "timeSlots/", params, config)
        .then((response) => {
          console.log(response);
          this.loadEducatorScheduleOverridesWithTimeSlots();
          this.$alertify.success("Créneau horaire ajouté avec succès");
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method to create a weekly schedule from the api rest endpoint "POST api/v1/weeklySchedules".
     *
     * @param {string} dateValidFrom The date valid from of the weekly schedule
     * @param {?string} dateValidTo The date valid to of the weekly schedule
     */
    createWeeklySchedule(dateValidFrom, dateValidTo) {
      const params = new URLSearchParams();
      params.append("date_valid_from", dateValidFrom);
      if (dateValidTo != null) {
        params.append("date_valid_to", dateValidTo);
      }
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token,
        },
      };
      this.$http
        .post(this.$API_URL + "weeklySchedules/", params, config)
        .then((response) => {
          console.log(response);
          this.loadEducatorWeeklySchedulesWithTimeSlots();
          this.$alertify.success("Calendrier hebdomadaire ajouté avec succès");
          this.$bvModal.hide("modal-add-weeklyschedule");
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method to create a schedule override from the api rest endpoint "POST api/v1/scheduleOverrides".
     *
     * @param {string} dateScheduleOverride The date of schedule override
     */
    createScheduleOverride(dateScheduleOverride) {
      const params = new URLSearchParams();
      params.append("date_schedule_override", dateScheduleOverride);
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token,
        },
      };
      this.$http
        .post(this.$API_URL + "scheduleOverrides/", params, config)
        .then((response) => {
          console.log(response);
          this.loadEducatorScheduleOverridesWithTimeSlots();
          this.$alertify.success("Exception d'horaire ajouté avec succès");
          this.$bvModal.hide("modal-add-scheduleoverride");
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method to create a absence from the api rest endpoint "POST api/v1/absences".
     *
     * @param {string} dateAbsenceFrom The date absence from of absence
     * @param {string} dateAbsenceTo The date absence to of absence
     */
    createAbsence(dateAbsenceFrom, dateAbsenceTo) {
      const params = new URLSearchParams();
      params.append("date_absence_from", dateAbsenceFrom);
      params.append("date_absence_to", dateAbsenceTo);
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token,
        },
      };
      this.$http
        .post(this.$API_URL + "absences/", params, config)
        .then((response) => {
          console.log(response);
          this.loadEducatorAbsences();
          this.$alertify.success("Vacances ajoutées avec succès");
          this.$bvModal.hide("modal-add-absence");
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method to delete a weekly schedule from his id from the api rest endpoint "DELETE api/v1/weeklySchedules/{weeklyScheduleId}"
     *
     * @param {string} id The weekly schedule id
     */
    deleteWeeklyScheduleById(id) {
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token,
        },
      };
      this.$http
        .delete(this.$API_URL + "weeklySchedules/" + id, config)
        .then((response) => {
          this.loadEducatorWeeklySchedulesWithTimeSlots();
          this.$alertify.success("Calendrier hebdomadaire supprimé");
          console.log(response);
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method to delete a schedule override from his id from the api rest endpoint "DELETE api/v1/scheduleOverrides/{scheduleOverrideId}"
     *
     * @param {string} id The schedule override id
     */
    deleteScheduleOverrideById(id) {
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token,
        },
      };
      this.$http
        .delete(this.$API_URL + "scheduleOverrides/" + id, config)
        .then((response) => {
          this.loadEducatorScheduleOverridesWithTimeSlots();
          this.$alertify.success("Exception d'horaire supprimé");
          console.log(response);
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method to delete a absence from his id from the api rest endpoint "DELETE api/v1/absences/{absenceId}"
     *
     * @param {string} id The absence id
     */
    deleteAbsenceById(id) {
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token,
        },
      };
      this.$http
        .delete(this.$API_URL + "absences/" + id, config)
        .then((response) => {
          this.loadEducatorAbsences();
          this.$alertify.success("Vacances supprimées");
          console.log(response);
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    /**
     * Method to delete a time slot from his id from the api rest endpoint "DELETE api/v1/timeSlots/{timeSlotId}"
     *
     * @param {string} id The time slot id
     */
    deleteTimeSlotById(timeSlotId) {
      const config = {
        headers: {
          "Authorization" : this.$store.state.api_token,
        },
      };
      this.$http
        .delete(this.$API_URL + "timeSlots/" + timeSlotId, config)
        .then((response) => {
          this.loadEducatorWeeklySchedulesWithTimeSlots();
          this.loadEducatorScheduleOverridesWithTimeSlots();
          this.$alertify.success("Créneau horaire supprimé");
          console.log(response);
        })
        .catch((error) => {
          this.$alertify.error(error.response.data.error);
        });
    },
    sendWeeklyScheduleId(weeklyScheduleId) {
      this.selectedWeeklyScheduleId = weeklyScheduleId;
    },
    sendScheduleOverrideId(scheduleOverrideId) {
      this.selectedScheduleOverrideId = scheduleOverrideId;
    },
    sendTimeSlotId(timeSlotId) {
      this.selectedTimeSlotId = timeSlotId;
    },
    sendAbsenceId(absenceId) {
      this.selectedAbsenceId = absenceId;
    },
    formatDate(value) {
      if (value) {
        return moment(value).format("dddd DD MMMM YYYY");
      }
    },
    formatTime(value) {
      if (value) {
        return moment("1970-01-01 " + value).format("HH[h]");
      }
    },
    clearWeeklyScheduleDateValidTo() {
      this.formWeeklySchedule.date_valid_to = null;
    },
  },
  mounted() {
    this.loadEducatorWeeklySchedulesWithTimeSlots();
    this.loadEducatorScheduleOverridesWithTimeSlots();
    this.loadEducatorAbsences();
  },
};
</script>

<style scoped>
.title {
  margin-top: 20px;
  color: #3ea3d8;
}
.tableBtn {
  background-color: #3ea3d8;
}
</style>
```

Listing 16 – ./Sources/src/components/PrivacyPolicy.vue

```
<!--
  PrivacyPolicy.vue

  Component representing the privacy policies of the application.

  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
-->

<template>
  <div class="content">
    <b-container>
      <b-row id="title" class="text-center">
        <b-col>
          <h1 class="font-weight-bold">Politique de condidentalité</h1>
        </b-col>
      </b-row>
      <b-row class="text-center">
        <b-col>
          <small
            ><p style="margin-bottom: 0px">
              Politique de confidentialité
            </p></small
          >
          <small><p>Dernière révision : 15/04/2021</p></small>
        </b-col>
      </b-row>
      <b-row>
        <b-col id="privacyPolicy">
          <small
```

```
><p>Introduction</p>
<p class="text-justify">
  Nous nous soucions de la vie privée des utilisateurs de notre site
  Internet et/ou de notre espace mobile (le « Site  et '«l Espace
  mobile , respectivement) et nous nous engageons à protéger les
  informations que les utilisateurs partagent avec nous 'lorsquils
  utilisent notre Site et/ou notre Espace mobile (collectivement,
  les « Propriétés numériques ), et nous nous engageons pleinement
  à protéger et à utiliser vos informations conformément à la loi
  applicable. Cette Politique de confidentialité décrit nos
  pratiques en matière de recueil, d'utilisation et de divulgation
  de vos informations par le biais de nos propriétés numériques (les«
   Services ), lorsque vous accédez aux Services depuis votre
  appareil. Avant d'accéder ou d'utiliser l'un de nos Services,
  veuillez lire attentivement cette Politique de confidentialité et
  vous assurer que vous comprenez parfaitement nos pratiques
  concernant vos informations. Si vous lisez et comprenez pleinement
  cette Politique de confidentialité, et que vous restez opposé(e) à
  nos pratiques, vous devez immédiatement et cesser toute
  utilisation de nos Propriétés numérique et de nos Services. En
  utilisant nos Services, vous acceptez les conditions de cette
  Politique de confidentialité et le fait de continuer à utiliser
  les Services constitue votre acceptation de cette Politique de
  confidentialité et de tout amendement à celle-ci. Dans la présente
  Politique de confidentialité, vous trouverez des explications sur
  les questions suivantes :
</p>
<ul>
  <li>Les informations que nous recueillons</li>
  <li>Comment nous recueillons les informations</li>
</ul>
<p>
  Les informations que nous recueillons<br />Vous trouverez
  ci-dessous les différents types d'informations que nous pouvons
  recueillir.
</p>
<ul>
  <li>
    Les informations non identifiées et non identifiables, qui
    peuvent être fournies par vous lors du processus
    d'enregistrement ou recueillies lorsque vous utilisez nos
    Services (les « Informations non personnelles ). Les
    informations non personnelles ne nous permettent pas
    d'identifier la personne auprès de qui elles ont été
    recueillies. Les Informations non personnelles que nous
    recueillons consistent principalement en des informations
    techniques et des informations d'utilisation agrégées.
  </li>
  <li>
    Les informations permettant de vous identifier individuellement,'
    cest-à-dire les informations qui vous identifient ou qui
    peuvent, moyennant un effort raisonnable, vous identifier (les «
    Informations personnelles ). Les Informations personnelles que
    nous recueillons par le biais de nos Services sont notamment le
    nom, l'adresse e-mail, l'adresse, le numéro de téléphone, ou
    d'autres informations demandées à chaque fois que nécessaire. Si
    nous combinons des Informations personnelles avec des
    Informations non personnelles, nous traiterons les informations
    combinées comme des Informations personnelles aussi longtemps
```

```
                longtemps qu’elles resteront combinées.
              </li>
            </ul>
            <p>
              Comment nous recueillons les informations<br />Vous trouverez
              ci-dessous les principales méthodes que nous utilisons pour
              recueillir des informations :
            </p>
            <ul>
              <li>
                Nous recueillons des informations lorsque vous utilisez les
                Services. Lorsque vous visitez nos Propriétés numériques ou que
                vous utilisez nos Services, nous pouvons recueillir, collecter
                et enregistrer ces utilisations, sessions et informations
                connexes.
              </li>
              <li>
                Nous recueillons les informations que vous nous fournissez
                volontairement. Par exemple, nous recueillons les informations
                que vous nous fournissez lorsque vous nous contactez directement
                par n’importe quel canal de communication (par exemple lorsque
                vous nous envoyez un e-mail contenant un commentaire ou un
                retour d’information).
              </li>
            </ul>
          </small>
        </b-col>
      </b-row>
    </b-container>
  </div>
</template>

<script>
export default {
  name: "PrivacyPolicy",
};
</script>

<style scoped>
#title {
  margin-top: 20px;
  color: #3ea3d8;
}
</style>
```

Listing 17 – ./Sources/src/components/RightSectionContent.vue

```
<!--
  RightSectionContent.vue

  Reusable component representing an aesthetic content displayed with textual content on the right and an image on the left.

  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
-->

<template>
  <div class="content">
    <b-container>
```

```
      <b-row align-h="center">
        <b-col
          class="d-flex align-items-center justify-content-center"
          cols="12"
          sm="12"
          lg="6"
        >
          <b-img
            v-if="imageSource != null"
            class="circleImage"
            :src="imageSource"
            fluid
            rounded="circle"
            alt="Circle image"
          ></b-img>
        </b-col>
        <b-col cols="12" sm="12" lg="6">
          <h1 class="font-weight-bold display-4">{{ textH1 }}</h1>
          <h3 class="font-weight-bold">{{ textH3 }}</h3>
          <h5>{{ textH5 }}</h5>
          <h6>{{ textH6 }}</h6>
          <p class="text-justify">{{ textParagraph }}</p>
          <b-button id="button" pill :to="routeBouton" v-if="textButton != ''">
            {{ textButton }}</b-button
          >
        </b-col>
      </b-row>
    </b-container>
  </div>
</template>

<script>
export default {
  props: {
    textParagraph: {
      type: String,
      default: "",
    },
    textH1: {
      type: String,
      default: "",
    },
    textH3: {
      type: String,
      default: "",
    },
    textH5: {
      type: String,
      default: "",
    },
    textH6: {
      type: String,
      default: "",
    },
    routeBouton: {
      type: String,
      default: "",
    },
    textButton: {
      type: String,
      default: "",
    },
    imageSource: {
      type: String,
      default: "",
    },
  },
  name: "RightSectionContent",
};
</script>

<style scoped>
.content {
  background-color: #ffffff;
  padding-top: 20px;
  padding-bottom: 20px;
  color: #3ea3d8;
}
#button {
  background-color: #3ea3d8;
}
</style>
```

Listing 18 – ./Sources/src/components/SignaturePad.vue

```
<!--
  PrivacyPolicy.vue

  Reusable component representing the signature functionality.

  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
-->

<template>
  <div class="content">
    <b-container id="container">
      <VueSignaturePad
        id="pad"
        :customStyle="customStyle"
        ref="signaturePad"
        :options="signaturePadOptions"
      />
      <b-row id="buttonRow" class="text-center">
        <b-col>
          <b-button
            id="btnClearSignature"
            block
            type="button"
            @click="clearSignaturePad"
            variant="outline-danger"
          >
            Effacer
          </b-button>
        </b-col>
        <b-col>
          <b-button
            id="btnValidateSignature"
            :disabled="isBlocked"
            block
            type="button"
            @click="getSignaturePad"
            variant="outline-success"
          >
            Valider la signature
          </b-button>
        </b-col>
      </b-row>
    </b-container>
  </div>
</template>

<script>
export default {
  name: "SignaturePad",
  data() {
    return {
      dataURL: "",
      isBlocked: false,
      customStyle: { border: "#c2c2c2 3px solid" },
      signaturePadOptions: {
        onBegin: () => {
          this.$refs.signaturePad.resizeCanvas();
        },
        backgroundColor: "rgb(255, 255, 255)",
      },
    };
  },
  methods: {
    getSignaturePad() {
      this.isBlocked = true;
      this.$refs.signaturePad.lockSignaturePad();
      this.dataURL = this.$refs.signaturePad.saveSignature("image/jpeg").data;
      this.$emit("getSignaturePad", this.dataURL, this.isBlocked);
    },
    clearSignaturePad() {
      this.$refs.signaturePad.clearSignature();
      this.isBlocked = false;
      this.$refs.signaturePad.openSignaturePad();
      this.$emit("getSignaturePad", this.dataURL, this.isBlocked);
    },
    resizeSignaturePadToSquare() {
      this.$refs.signaturePad.resizeCanvas();
      var padWidth = this.$jquery("#pad").width();
      this.$jquery("#pad").css({ height: padWidth + "px" });
    },
  },
  destroyed() {
    window.removeEventListener("resize", this.resizeSignaturePadToSquare);
  },
  mounted() {
    window.addEventListener("resize", this.resizeSignaturePadToSquare);
    this.resizeSignaturePadToSquare();
  },
};
</script>

<style scoped>
```

```
#container {
  padding: 0px;
}
#buttonRow {
  margin-top: 10px;
  margin-bottom: 10px;
}
</style>
```

Listing 19 – ./Sources/src/constants.js

```
/*
  constants.js

  File containing the constants of the application.

  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
*/

export const CUSTOMER_CODE_ROLE = "1";
export const ADMIN_CODE_ROLE = "2";
```

Listing 20 – ./Sources/src/main.js

```
/*
  main.js

  File allowing to initialize the main root component with all the configuration and the plugins necessary for the good functioning of the application.

  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
*/

import "@babel/polyfill";
import "mutationobserver-shim";

import "./plugins/bootstrap-vue";
import "./plugins/vue-alertify";
import "./plugins/vue-signature-pad";

import "./assets/css/main.css";

import Vue from "vue";
import App from "./App.vue";
import router from "./router";
import store from "./store";

import Axios from "axios";
Vue.prototype.$http = Axios;
Vue.prototype.$API_URL = "https://api-rest-douceur-de-chien.boreljaquet.ch/v1/";

import jquery from "jquery";
Vue.prototype.$jquery = jquery;

new Vue({
  router,
  store,
  render: (h) => h(App),
}).$mount("#app");
```

Listing 21 – ./Sources/src/plugins/bootstrap-vue.js

```
/*
  bootstrap-vue.js

  File allowing to include all the files necessary to the good functioning of bootstrapVue in Vue.js.

  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
*/

import Vue from "vue";

import BootstrapVue from "bootstrap-vue";
import "bootstrap/dist/css/bootstrap.min.css";
import "bootstrap-vue/dist/bootstrap-vue.css";

Vue.use(BootstrapVue);
```

Listing 22 – ./Sources/src/plugins/vue-alertify.js

```
/*
  vue-alertify.js

  File allowing to include all the files necessary to the good functioning of vue-alertify in Vue.js.

  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
*/

import Vue from "vue";

import VueAlertify from "vue-alertify";
Vue.use(VueAlertify);
```

Listing 23 – ./Sources/src/plugins/vue-signature-pad.js

```
/*
  vue-signature-pad.js

  File allowing to include all the files necessary to the good functioning of vue-signature-pad in Vue.js.

  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
*/

import Vue from "vue";

import VueSignaturePad from "vue-signature-pad";
Vue.use(VueSignaturePad);
```

Listing 24 – ./Sources/src/router.js

```
/*
  router.js

  File to configure the application router with Vue Router.

  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
*/
```

```

import Vue from "vue";
import Router from "vue-router";
import store from "./store";
import Home from "./components/Home.vue";
import About from "./components/About.vue";
import Calendar from "./components/Calendar.vue";
import PrivacyPolicy from "./components/PrivacyPolicy.vue";
import Connection from "./components/Connection.vue";
import Inscription from "./components/Inscription.vue";
import CustomerInformation from "./components/CustomerInformation.vue";
import CustomerAppointment from "./components/CustomerAppointment.vue";
import Administration from "./components/Administration.vue";
import Planning from "./components/Planning.vue";
import EducatorCalendar from "./components/EducatorCalendar.vue";
import { CUSTOMER_CODE_ROLE, ADMIN_CODE_ROLE } from "./constants.js";

Vue.use(Router);

export default new Router({
  routes: [
    { path: "/", name: "home", component: Home },
    { path: "/about", name: "about", component: About },
    { path: "/calendar", name: "calendar", component: Calendar },
    {
      path: "/privacy_policy",
      name: "privacyPolicy",
      component: PrivacyPolicy,
    },
    {
      path: "/connection",
      name: "connection",
      component: Connection,
      beforeEnter(to, from, next) {
        if (
          store.state.api_token &&
          store.state.code_role == CUSTOMER_CODE_ROLE
        ) {
          next("/customer_information");
        } else {
          next();
        }
        if (store.state.api_token && store.state.code_role == ADMIN_CODE_ROLE) {
          next("/administration");
        } else {
          next();
        }
      },
    },
    {
      path: "/inscription",
      name: "inscription",
      component: Inscription,
      beforeEnter(to, from, next) {
        if (
          store.state.api_token &&
          store.state.code_role == CUSTOMER_CODE_ROLE
        ) {
          next("/customer_information");
        } else {
          next();
        }
        if (store.state.api_token && store.state.code_role == ADMIN_CODE_ROLE) {
          next("/administration");
        } else {
          next();
        }
      },
    },
    {
      path: "/customer_information/:userId?",
      name: "customerInformation",
      component: CustomerInformation,
      beforeEnter(to, from, next) {
        if (store.state.api_token) {
          next();
        } else {
          next("/");
        }
      },
    },
    {
      path: "/customer_appointment/:userId?",
      name: "customerAppointment",
      component: CustomerAppointment,
      beforeEnter(to, from, next) {
        if (store.state.api_token) {
          next();
        } else {
          next("/");
        }
      },
    },
    {
      path: "/administration",
      name: "administration",
      component: Administration,
      beforeEnter(to, from, next) {
        if (store.state.api_token && store.state.code_role == ADMIN_CODE_ROLE) {
          next();
        } else {
          next("/");
        }
      },
    },
    {
      path: "/planning",
      name: "planning",
      component: Planning,
      beforeEnter(to, from, next) {
        if (store.state.api_token && store.state.code_role == ADMIN_CODE_ROLE) {
          next();
        } else {
          next("/");
        }
      },
    },
    {
      path: "/educator_calendar",
      name: "educatorCalendar",
      component: EducatorCalendar,
      beforeEnter(to, from, next) {
        if (store.state.api_token && store.state.code_role == ADMIN_CODE_ROLE) {
          next();
        } else {
          next("/");
        }
      },
    },
  ],
});
```

Listing 25 – ./Sources/src/store.js

```
/*
  store.js

  File to configure the application store with Vuex.

  Jonathan Borel-Jaquet - CFPT / T.IS-ES2 <jonathan.brljq@eduge.ch>
*/

import Vue from "vue";
import Vuex from "vuex";
import createPersistedState from "vuex-persistedstate";
import router from "./router";
import { CUSTOMER_CODE_ROLE, ADMIN_CODE_ROLE } from "./constants.js";

Vue.use(Vuex);

export default new Vuex.Store({
  plugins: [createPersistedState()],
  state: {
    api_token: null,
    code_role: null,
    user_id: null,
  },
  mutations: {
    authUser(state, userData) {
      state.api_token = userData.api_token;
      state.code_role = userData.code_role;
      state.user_id = userData.user_id;
    },
    clearAuth(state) {
      state.api_token = null;
      state.code_role = null;
      state.user_id = null;
    },
  },
  actions: {
    inscription({ commit }, authData) {
      const params = new URLSearchParams();
      params.append("email", authData.email);
      params.append("firstname", authData.firstname);
      params.append("lastname", authData.lastname);
      params.append("phonenumber", authData.phonenumber);
      params.append("address", authData.address);
      params.append("password", authData.password);
      params.append(
        "reCAPTCHAuserResponseToken",
        authData.reCAPTCHAuserResponseToken
      );
      Vue.prototype.$http
        .post(Vue.prototype.$API_URL + "users/", params)
        .then((res) => {
          console.log(res);
          localStorage.setItem("api_token", res.data.api_token);
          localStorage.setItem("code_role", res.data.code_role);
          localStorage.setItem("user_id", res.data.user_id);
          commit("authUser", {
            api_token: res.data.api_token,
            code_role: res.data.code_role,
            user_id: res.data.user_id,
          });
          Vue.prototype.$alertify.success("Vous êtes inscrit");
          router.push("/customer_information");
        })
        .catch((error) => {
          Vue.prototype.$alertify.error(error.response.data.error);
        });
    },
    login({ commit }, authData) {
      const params = new URLSearchParams();
      params.append("email", authData.email);
      params.append("password", authData.password);
      Vue.prototype.$http
        .post(Vue.prototype.$API_URL + "connection/", params)
        .then((res) => {
          localStorage.setItem("api_token", res.data.api_token);
          localStorage.setItem("code_role", res.data.code_role);
          localStorage.setItem("user_id", res.data.user_id);
          commit("authUser", {
            api_token: res.data.api_token,
            code_role: res.data.code_role,
            user_id: res.data.user_id,
          });
          Vue.prototype.$alertify.success("Vous êtes connecté");
          if (this.state.code_role == ADMIN_CODE_ROLE) {
            router.push("/administration");
          }
          if (this.state.code_role == CUSTOMER_CODE_ROLE) {
            router.push("/customer_information");
          }
        })
        .catch((error) => {
          Vue.prototype.$alertify.error(error.response.data.error);
        });
    },
    logout({ commit }) {
      commit("clearAuth");
      localStorage.removeItem("api_token");
      localStorage.removeItem("code_role");
      localStorage.removeItem("user_id");
      Vue.prototype.$alertify.success("Vous êtes déconnecté");
      router.push("/connection");
    },
    autoLogin({ commit }) {
      const api_token = localStorage.getItem("api_token");
      const code_role = localStorage.getItem("code_role");
      const user_id = localStorage.getItem("user_id");
      if (!api_token || !code_role || !user_id) {
        return;
      }
      commit("authUser", {
        api_token: api_token,
        code_role: code_role,
        user_id: user_id,
      });
    },
  },
  getters: {
    ifCustomerAuthenticated(state) {
      return state.api_token !== null && state.code_role == CUSTOMER_CODE_ROLE;
    },
    ifAdministratorAuthenticated(state) {
      return state.api_token !== null && state.code_role == ADMIN_CODE_ROLE;
    },
    ifAuthenticated(state) {
      return state.api_token !== null;
    },
  },
});
```