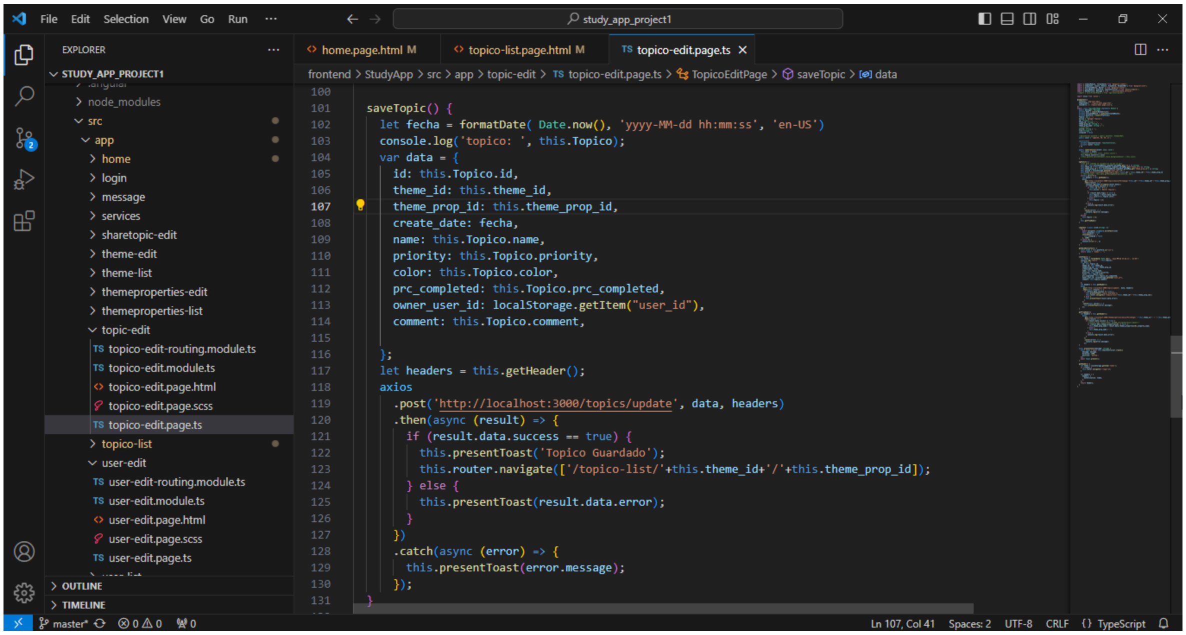
Robert Elias Leguizamon Gonzalez

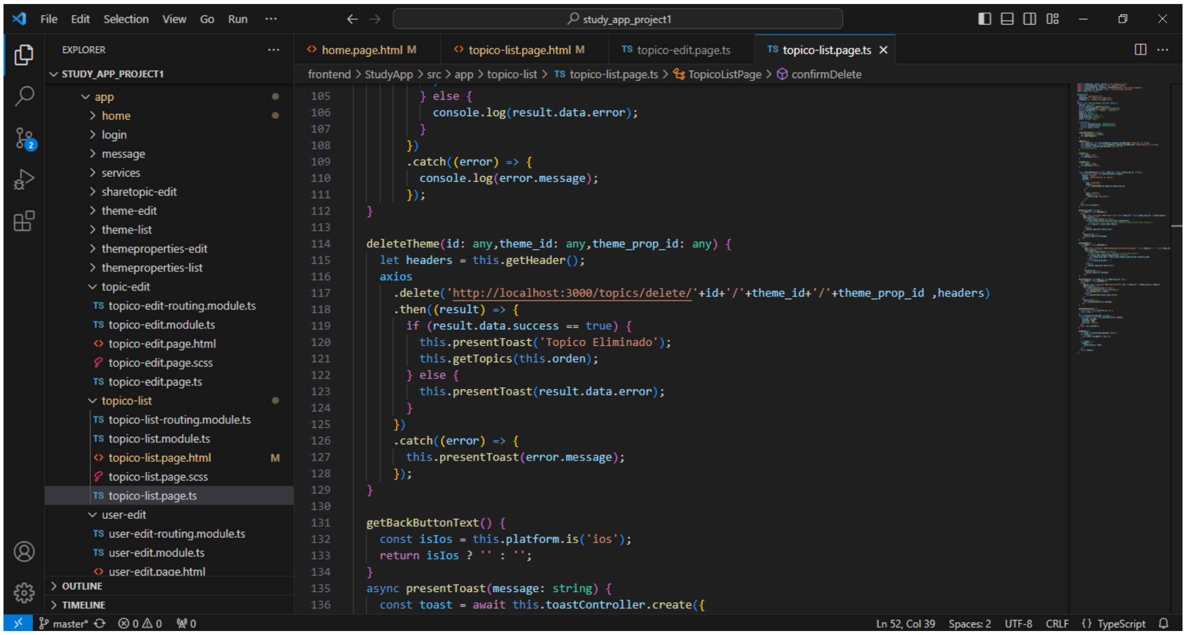
5939456

Examen Final

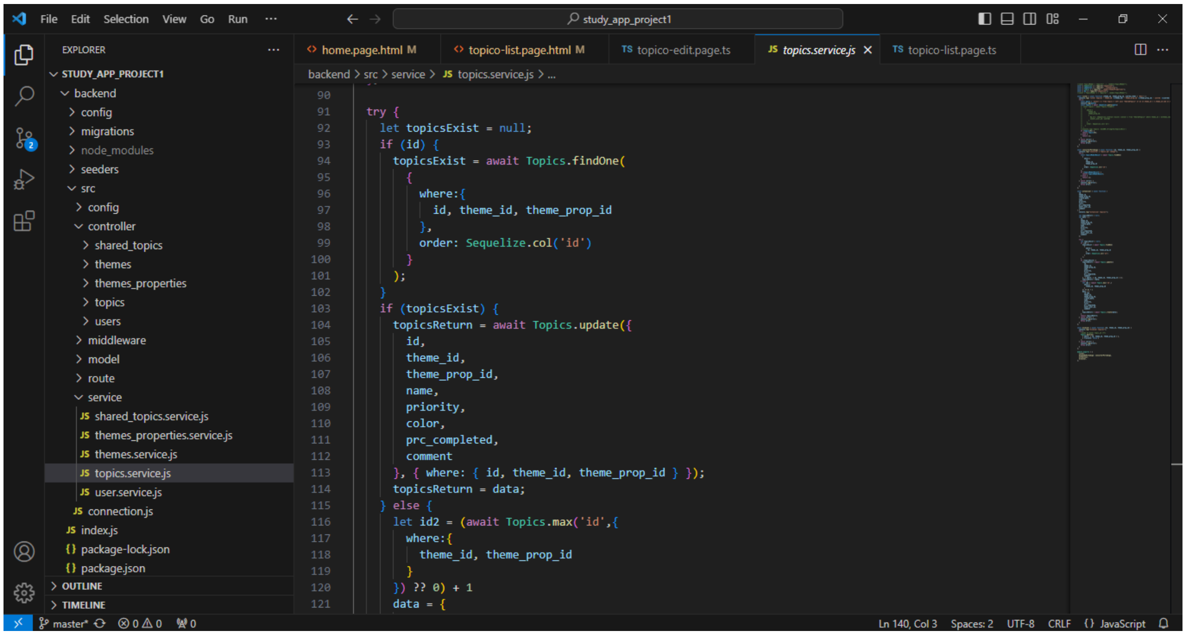
TOPICOS


home.page.html M
topico-list.page.html M
topico-edit.page.ts
frontend > StudyApp > src > app > topic-edit > topico-edit.page.ts > TopicoEditPage > saveTopic > data

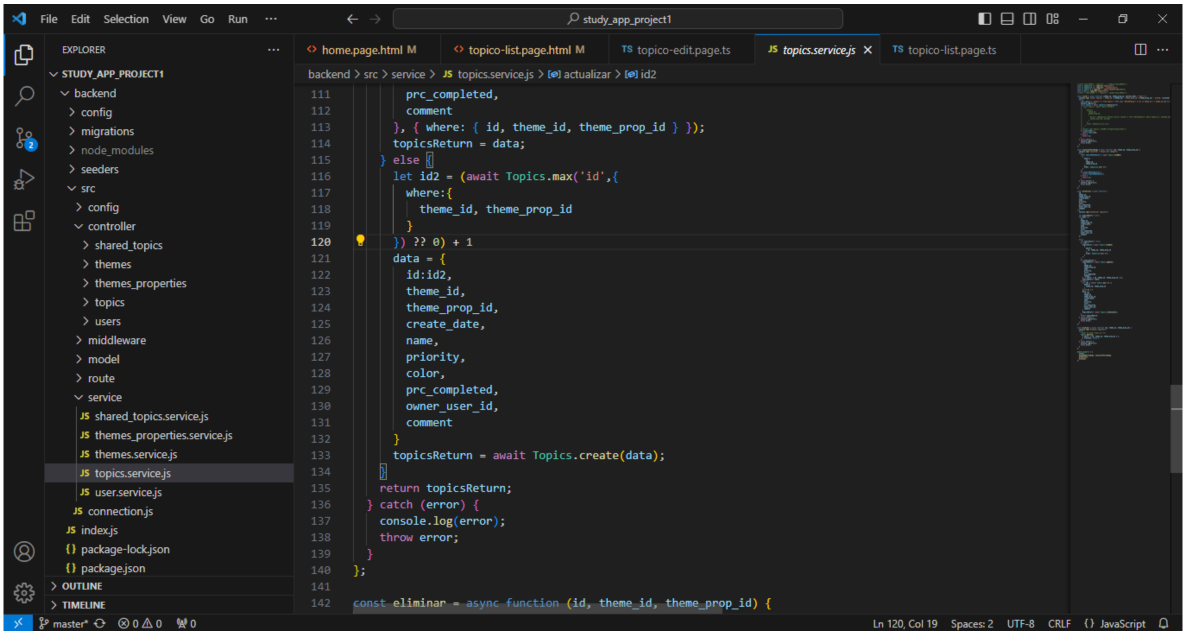
```
  saveTopic() {
    let fecha = formatDate( Date.now(), 'yyyy-MM-dd hh:mm:ss', 'en-US')
    console.log('topico: ', this.Topico);
    var data = {
      id: this.Topico.id,
      theme_id: this.theme_id,
      theme_prop_id: this.theme_prop_id,
      create_date: fecha,
      name: this.Topico.name,
      priority: this.Topico.priority,
      color: this.Topico.color,
      prc_completed: this.Topico.prc_completed,
      owner_user_id: localStorage.getItem("user_id"),
      comment: this.Topico.comment,

    };
    let headers = this.getHeader();
    axios
      .post('http://localhost:3000/topics/update', data, headers)
      .then(async (result) => {
        if (result.data.success == true) {
          this.presentToast('Topico Guardado');
          this.router.navigate(['/topico-list/'+this.theme_id+'/'+this.theme_prop_id]);
        } else {
          this.presentToast(result.data.error);
        }
      })
      .catch(async (error) => {
        this.presentToast(error.message);
      });
  }
```


home.page.html M
topico-list.page.html M
topico-edit.page.ts
topico-list.page.ts
frontend > StudyApp > src > app > topico-list > topico-list.page.ts > TopicoListPage > confirmDelete

```
          } else {
            console.log(result.data.error);
          }
        })
        .catch((error) => {
          console.log(error.message);
        });
  }

  deleteTheme(id: any,theme_id: any,theme_prop_id: any) {
    let headers = this.getHeader();
    axios
      .delete('http://localhost:3000/topics/delete/'+id+'/'+theme_id+'/'+theme_prop_id ,headers)
      .then((result) => {
        if (result.data.success == true) {
          this.presentToast('Topico Eliminado');
          this.getTopics(this.orden);
        } else {
          this.presentToast(result.data.error);
        }
      })
      .catch((error) => {
        this.presentToast(error.message);
      });
  }

  getBackButtonText() {
    const isIos = this.platform.is('ios');
    return isIos ? '' : '';
  }
  async presentToast(message: string) {
    const toast = await this.toastController.create({
```

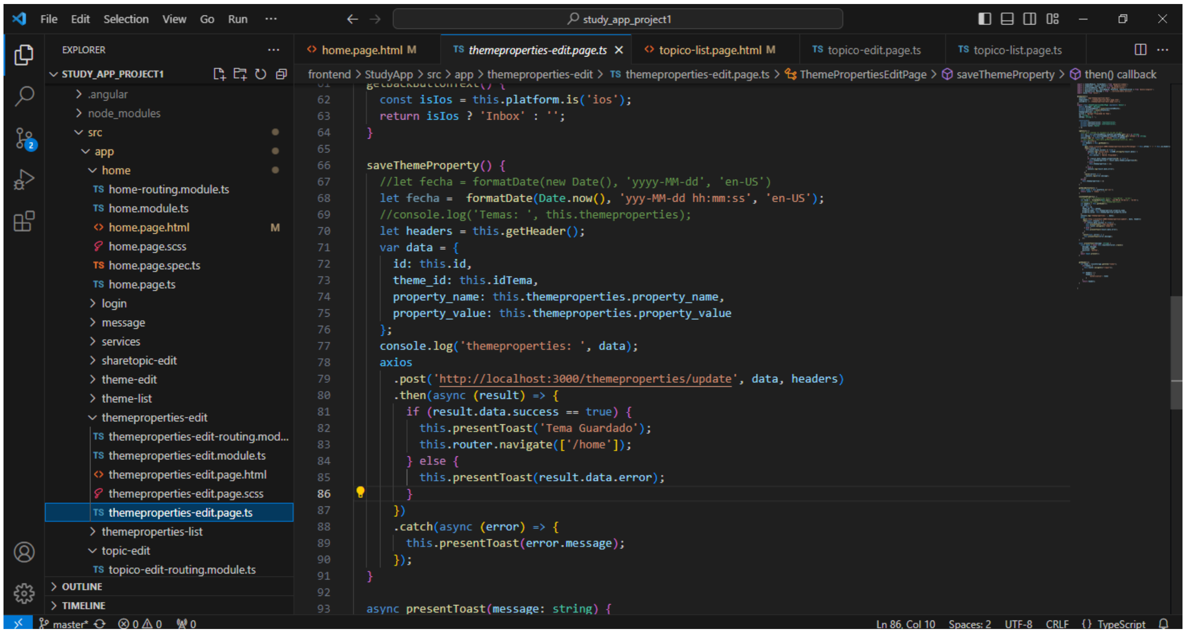


```
  try {
    let topicsExist = null;
    if (id) {
      topicsExist = await Topics.findOne(
        {
          where:{
            id, theme_id, theme_prop_id
          },
          order: Sequelize.col('id')
        }
      );
    }
    if (topicsExist) {
      topicsReturn = await Topics.update({
        id,
        theme_id,
        theme_prop_id,
        name,
        priority,
        color,
        prc_completed,
        comment
      }, { where: { id, theme_id, theme_prop_id } });
      topicsReturn = data;
    } else {
      let id2 = (await Topics.max('id',{
        where:{
          theme_id, theme_prop_id
        }
      }) ?? 0) + 1
      data = {
```

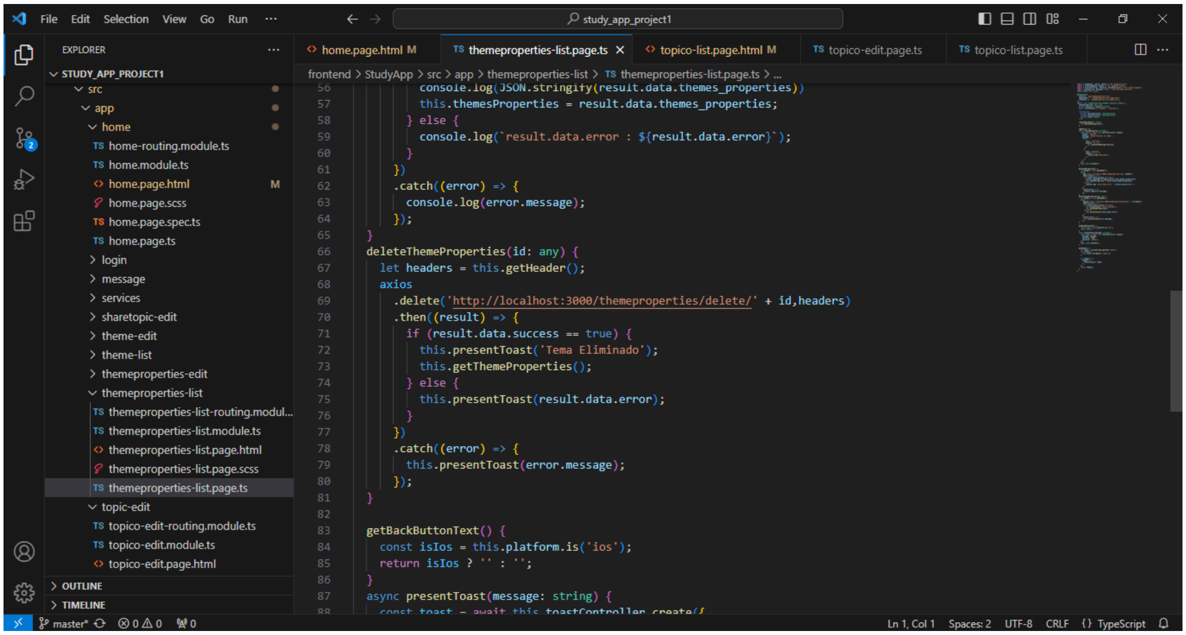


```
        prc_completed,
        comment
      }, { where: { id, theme_id, theme_prop_id } });
      topicsReturn = data;
    } else {
      let id2 = (await Topics.max('id',{
        where:{
          theme_id, theme_prop_id
        }
      }) ?? 0) + 1
      data = {
        id:id2,
        theme_id,
        theme_prop_id,
        create_date,
        name,
        priority,
        color,
        prc_completed,
        owner_user_id,
        comment
      }
      topicsReturn = await Topics.create(data);
    }
    return topicsReturn;
  } catch (error) {
    console.log(error);
    throw error;
  }
};

const eliminar = async function (id, theme_id, theme_prop_id) {
```

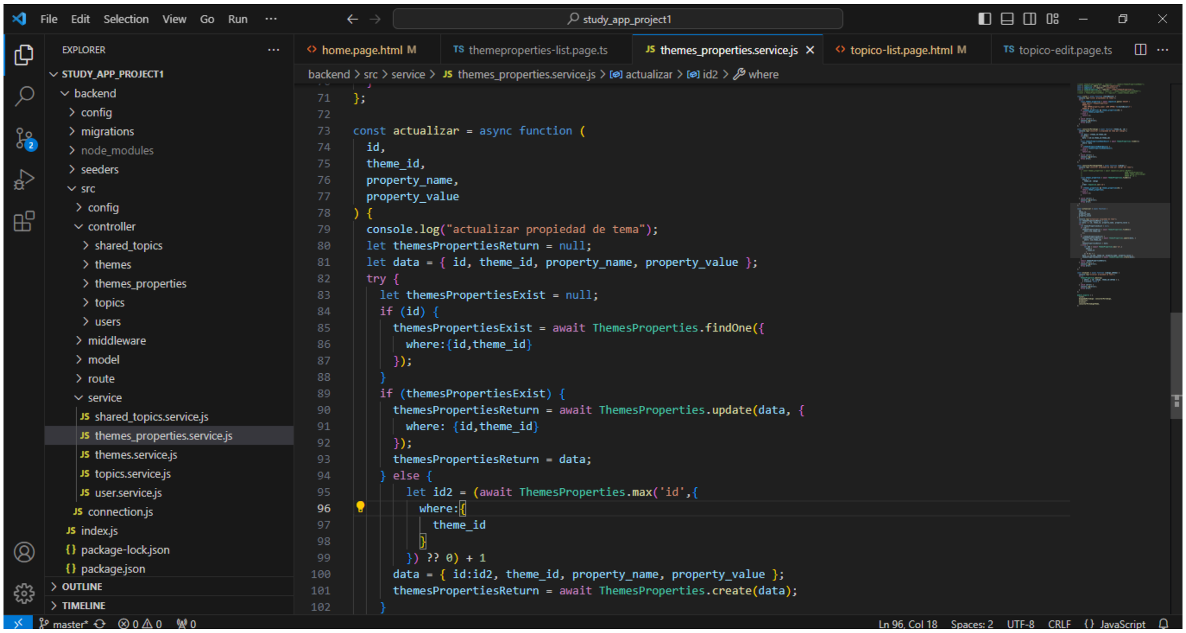
THEMES PROPERTIES



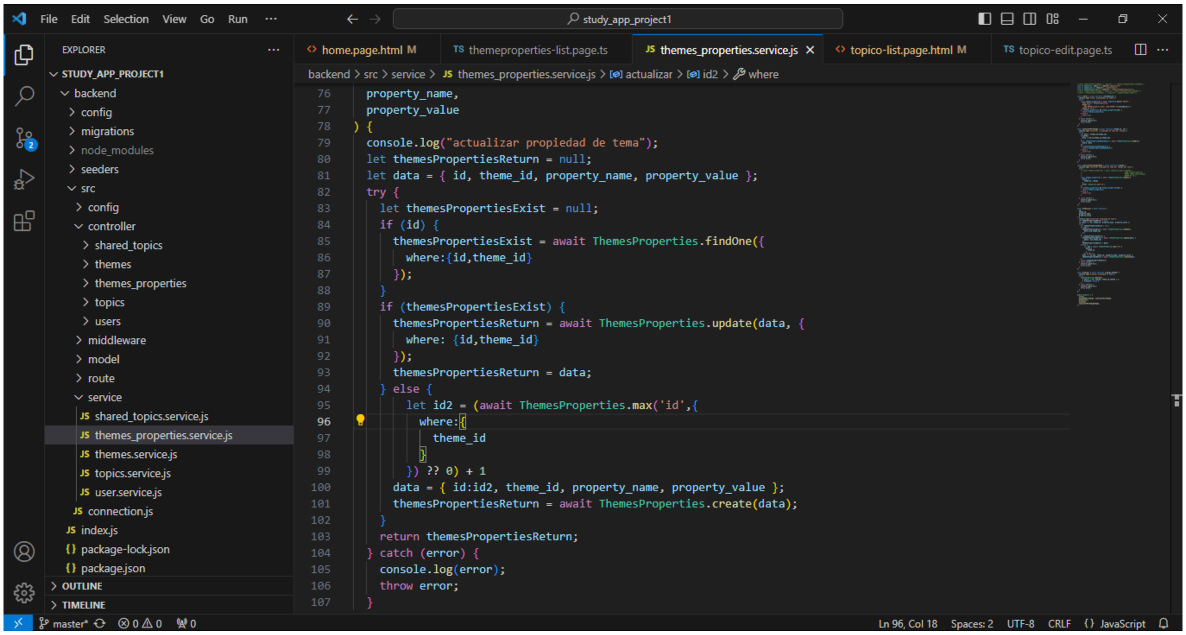
```
  getBackButtonText() {
    const isIos = this.platform.is('ios');
    return isIos ? 'Inbox' : '';
  }

  saveThemeProperty() {
    //let fecha = formatDate(new Date(), 'yyyy-MM-dd', 'en-US')
    let fecha =  formatDate(Date.now(), 'yyy-MM-dd hh:mm:ss', 'en-US');
    //console.log('Temas: ', this.themeproperties);
    let headers = this.getHeader();
    var data = {
      id: this.id,
      theme_id: this.idTema,
      property_name: this.themeproperties.property_name,
      property_value: this.themeproperties.property_value
    };
    console.log('themeproperties: ', data);
    axios
      .post('http://localhost:3000/themeproperties/update', data, headers)
      .then(async (result) => {
        if (result.data.success == true) {
          this.presentToast('Tema Guardado');
          this.router.navigate(['/home']);
        } else {
          this.presentToast(result.data.error);
        }
      })
      .catch(async (error) => {
        this.presentToast(error.message);
      });
  }

  async presentToast(message: string) {
```



```
          console.log(JSON.stringify(result.data.themes_properties))
          this.themesProperties = result.data.themes_properties;
        } else {
          console.log(`result.data.error : ${result.data.error}`);
        }
      })
      .catch((error) => {
        console.log(error.message);
      });
  }
  deleteThemeProperties(id: any) {
    let headers = this.getHeader();
    axios
      .delete('http://localhost:3000/themeproperties/delete/' + id,headers)
      .then((result) => {
        if (result.data.success == true) {
          this.presentToast('Tema Eliminado');
          this.getThemeProperties();
        } else {
          this.presentToast(result.data.error);
        }
      })
      .catch((error) => {
        this.presentToast(error.message);
      });
  }

  getBackButtonText() {
    const isIos = this.platform.is('ios');
    return isIos ? '' : '';
  }
  async presentToast(message: string) {
    const toast = await this.toastController.create({
```

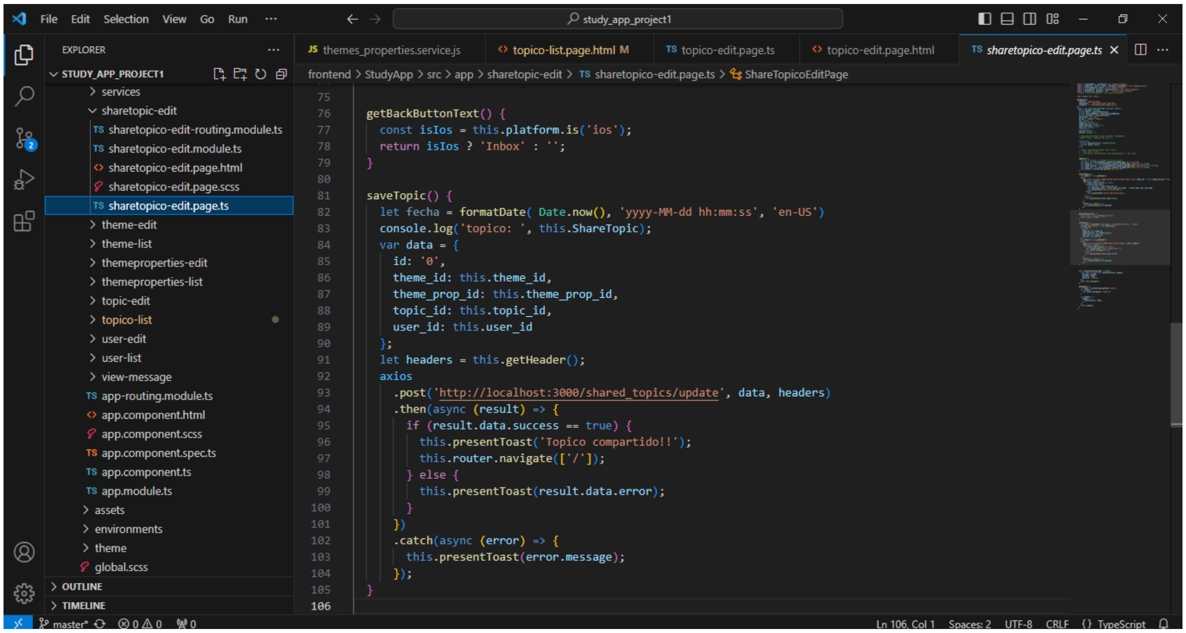


```
};

const actualizar = async function (
  id,
  theme_id,
  property_name,
  property_value
) {
  console.log("actualizar propiedad de tema");
  let themesPropertiesReturn = null;
  let data = { id, theme_id, property_name, property_value };
  try {
    let themesPropertiesExist = null;
    if (id) {
      themesPropertiesExist = await ThemesProperties.findOne({
        where:{id,theme_id}
      });
    }
    if (themesPropertiesExist) {
      themesPropertiesReturn = await ThemesProperties.update(data, {
        where: {id,theme_id}
      });
      themesPropertiesReturn = data;
    } else {
      let id2 = (await ThemesProperties.max('id',{
        where:{
          theme_id
        }
      }) ?? 0) + 1
      data = { id:id2, theme_id, property_name, property_value };
      themesPropertiesReturn = await ThemesProperties.create(data);
    }
```

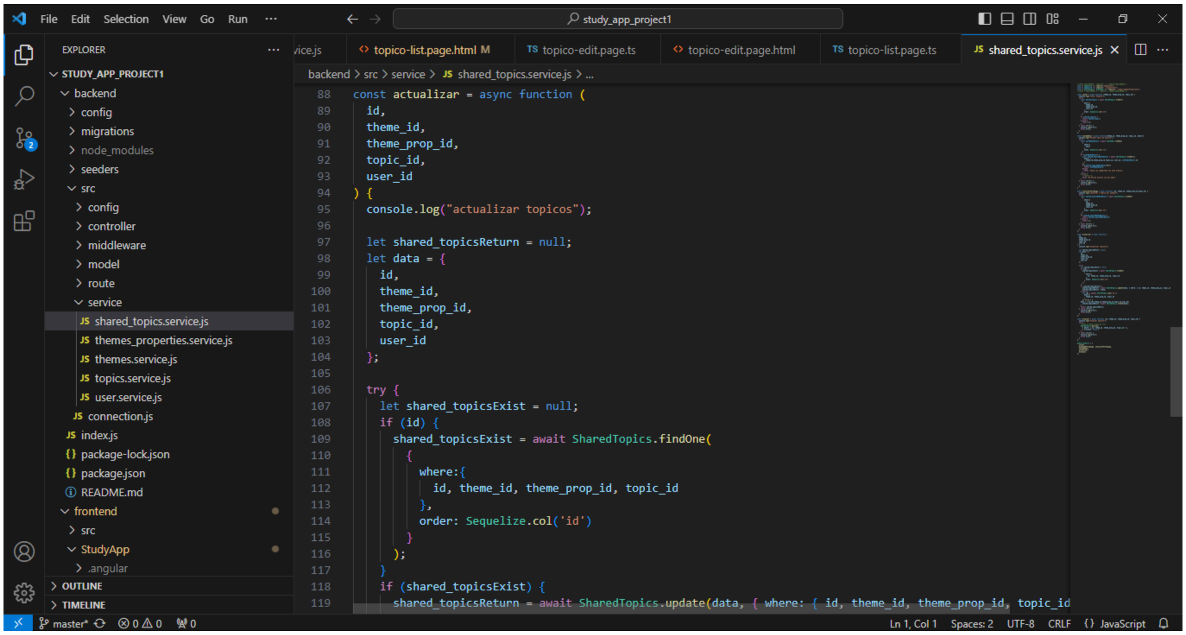


```
  property_name,
  property_value
) {
  console.log("actualizar propiedad de tema");
  let themesPropertiesReturn = null;
  let data = { id, theme_id, property_name, property_value };
  try {
    let themesPropertiesExist = null;
    if (id) {
      themesPropertiesExist = await ThemesProperties.findOne({
        where:{id,theme_id}
      });
    }
    if (themesPropertiesExist) {
      themesPropertiesReturn = await ThemesProperties.update(data, {
        where: {id,theme_id}
      });
      themesPropertiesReturn = data;
    } else {
      let id2 = (await ThemesProperties.max('id',{
        where:{
          theme_id
        }
      }) ?? 0) + 1
      data = { id:id2, theme_id, property_name, property_value };
      themesPropertiesReturn = await ThemesProperties.create(data);
    }
    return themesPropertiesReturn;
  } catch (error) {
    console.log(error);
    throw error;
  }
```

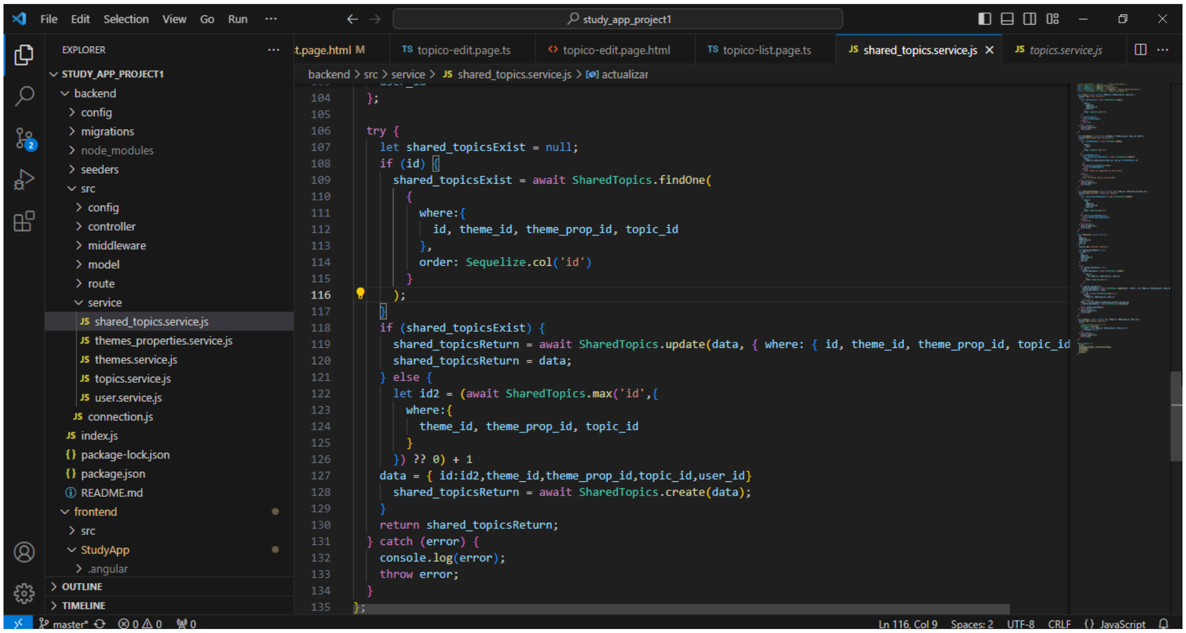
SHARED TOPICOS



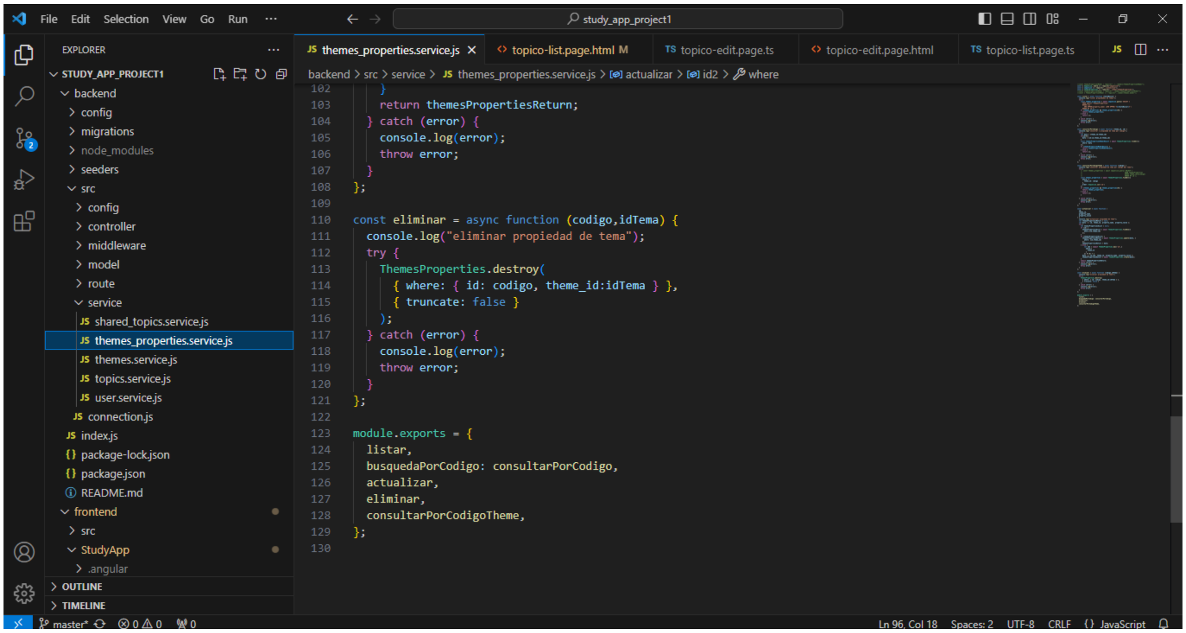
```
  getBackButtonText() {
    const isIos = this.platform.is('ios');
    return isIos ? 'Inbox' : '';
  }

  saveTopic() {
    let fecha = formatDate( Date.now(), 'yyyy-MM-dd hh:mm:ss', 'en-US')
    console.log('topico: ', this.ShareTopic);
    var data = {
      id: '0',
      theme_id: this.theme_id,
      theme_prop_id: this.theme_prop_id,
      topic_id: this.topic_id,
      user_id: this.user_id
    };
    let headers = this.getHeader();
    axios
      .post('http://localhost:3000/shared_topics/update', data, headers)
      .then(async (result) => {
        if (result.data.success == true) {
          this.presentToast('Topico compartido!!');
          this.router.navigate(['/']);
        } else {
          this.presentToast(result.data.error);
        }
      })
      .catch(async (error) => {
        this.presentToast(error.message);
      });
  }

```



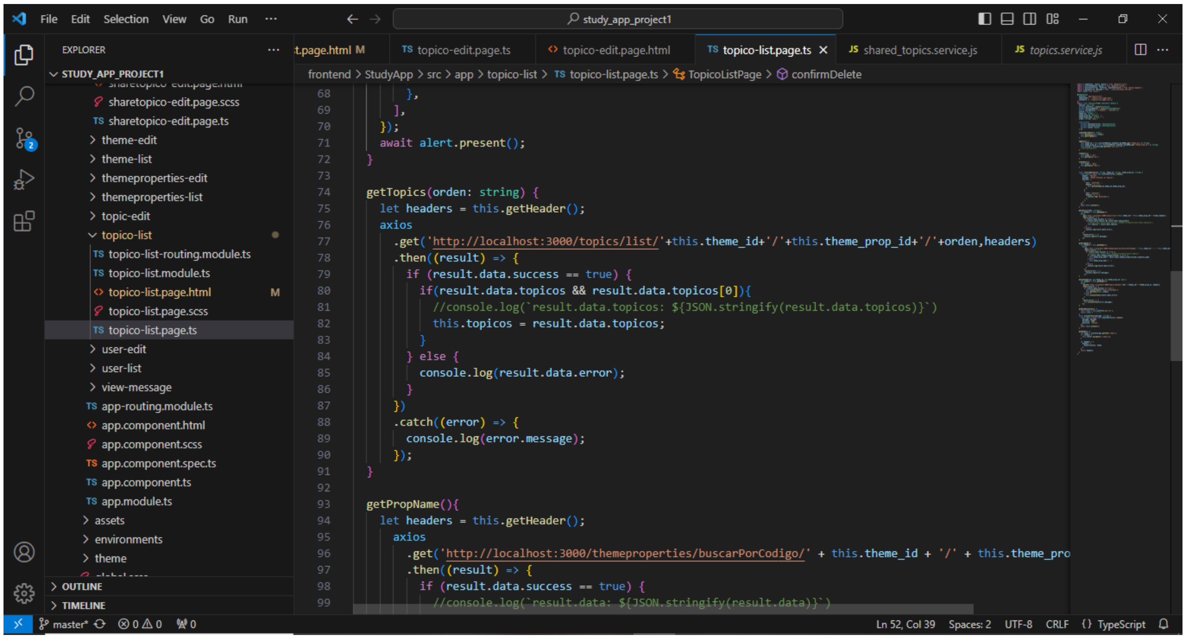
```
const actualizar = async function (
  id,
  theme_id,
  theme_prop_id,
  topic_id,
  user_id
) {
  console.log("actualizar topicos");

  let shared_topicsReturn = null;
  let data = {
    id,
    theme_id,
    theme_prop_id,
    topic_id,
    user_id
  };

  try {
    let shared_topicsExist = null;
    if (id) {
      shared_topicsExist = await SharedTopics.findOne(
        {
          where:{
            id, theme_id, theme_prop_id, topic_id
          },
          order: Sequelize.col('id')
        }
      );
    }
    if (shared_topicsExist) {
      shared_topicsReturn = await SharedTopics.update(data, { where: { id, theme_id, theme_prop_id, topic_id
```



```
};

try {
  let shared_topicsExist = null;
  if (id) {
    shared_topicsExist = await SharedTopics.findOne(
      {
        where:{
          id, theme_id, theme_prop_id, topic_id
        },
        order: Sequelize.col('id')
      }
    );
  }
  if (shared_topicsExist) {
    shared_topicsReturn = await SharedTopics.update(data, { where: { id, theme_id, theme_prop_id, topic_id
    shared_topicsReturn = data;
  } else {
    let id2 = (await SharedTopics.max('id',{
      where:{
        theme_id, theme_prop_id, topic_id
      }
    }) ?? 0) + 1
  data = { id:id2,theme_id,theme_prop_id,topic_id,user_id}
    shared_topicsReturn = await SharedTopics.create(data);
  }
  return shared_topicsReturn;
} catch (error) {
  console.log(error);
  throw error;
}
};
```

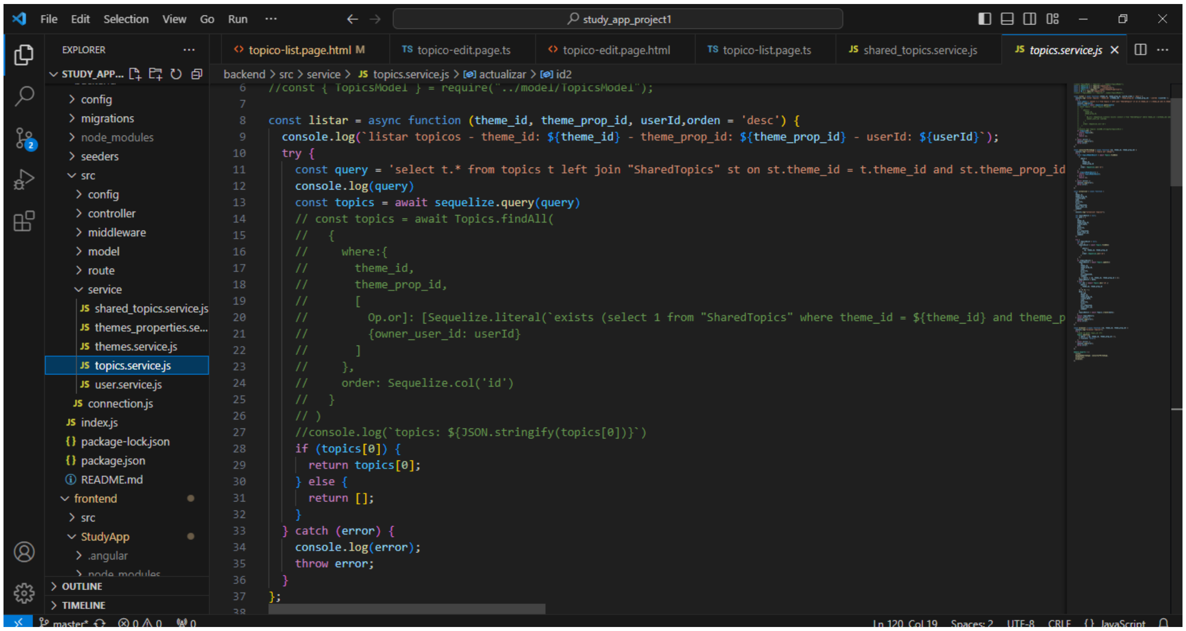


```
    }
    return themesPropertiesReturn;
  } catch (error) {
    console.log(error);
    throw error;
  }
};

const eliminar = async function (codigo,idTema) {
  console.log("eliminar propiedad de tema");
  try {
    ThemesProperties.destroy(
      { where: { id: codigo, theme_id:idTema } },
      { truncate: false }
    );
  } catch (error) {
    console.log(error);
    throw error;
  }
};

module.exports = {
  listar,
  busquedaPorCodigo: consultarPorCodigo,
  actualizar,
  eliminar,
  consultarPorCodigoTheme,
};
```



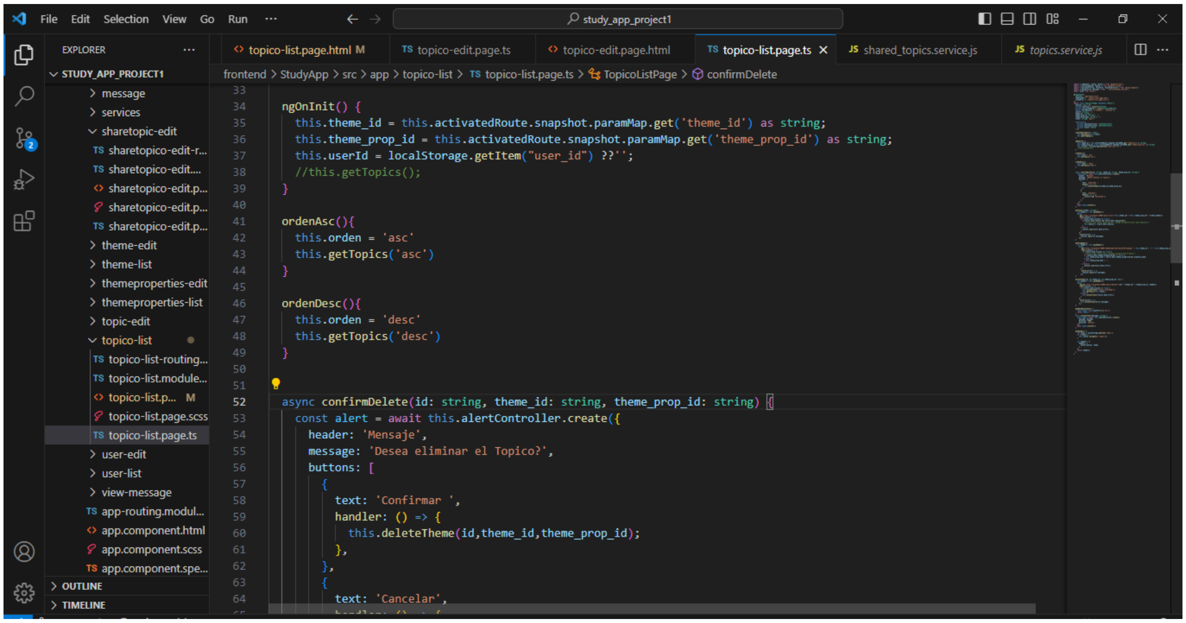
```
        },
      ],
    });
    await alert.present();
  }

  getTopics(orden: string) {
    let headers = this.getHeader();
    axios
      .get('http://localhost:3000/topics/list/'+this.theme_id+'/'+this.theme_prop_id+'/'+orden,headers)
      .then((result) => {
        if (result.data.success == true) {
          if(result.data.topicos && result.data.topicos[0]){
            //console.log(`result.data.topicos: ${JSON.stringify(result.data.topicos)}`)
            this.topicos = result.data.topicos;
          }
        } else {
          console.log(result.data.error);
        }
      })
      .catch((error) => {
        console.log(error.message);
      });
  }

  getPropName(){
    let headers = this.getHeader();
      axios
        .get('http://localhost:3000/themeproperties/buscarPorCodigo/' + this.theme_id + '/' + this.theme_pro
        .then((result) => {
          if (result.data.success == true) {
            //console.log(`result.data: ${JSON.stringify(result.data)}`)
```

VER TOPICOS COMPARTIDOS

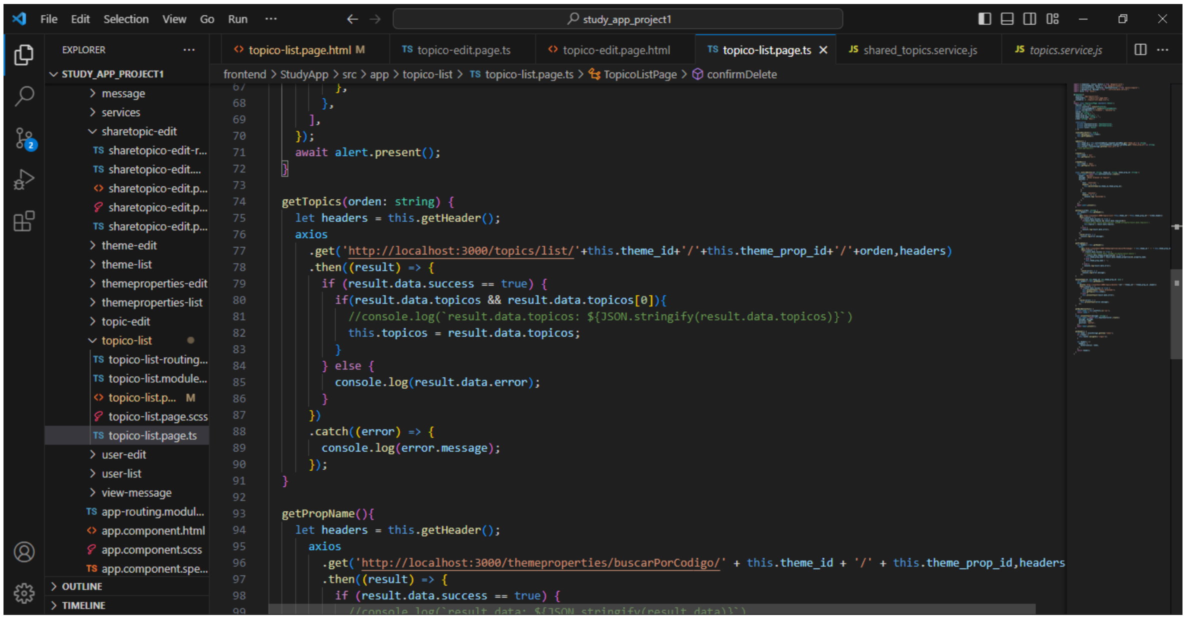


```
//const { TopicsModel } = require("../model/TopicsModel");

const listar = async function (theme_id, theme_prop_id, userId,orden = 'desc') {
  console.log(`listar topicos - theme_id: ${theme_id} - theme_prop_id: ${theme_prop_id} - userId: ${userId}`);
  try {
    const query = 'select t.* from topics t left join "SharedTopics" st on st.theme_id = t.theme_id and st.theme_prop_id
    console.log(query)
    const topics = await sequelize.query(query)
    // const topics = await Topics.findAll(
    //   {
    //     where:{
    //       theme_id,
    //       theme_prop_id,
    //       [
    //         Op.or]: [Sequelize.literal(`exists (select 1 from "SharedTopics" where theme_id = ${theme_id} and theme_p
    //         {owner_user_id: userId}
    //       ]
    //     },
    //     order: Sequelize.col('id')
    //   }
    // )
    //console.log(`topics: ${JSON.stringify(topics[0])}`)
    if (topics[0]) {
      return topics[0];
    } else {
      return [];
    }
  } catch (error) {
    console.log(error);
    throw error;
  }
};
```

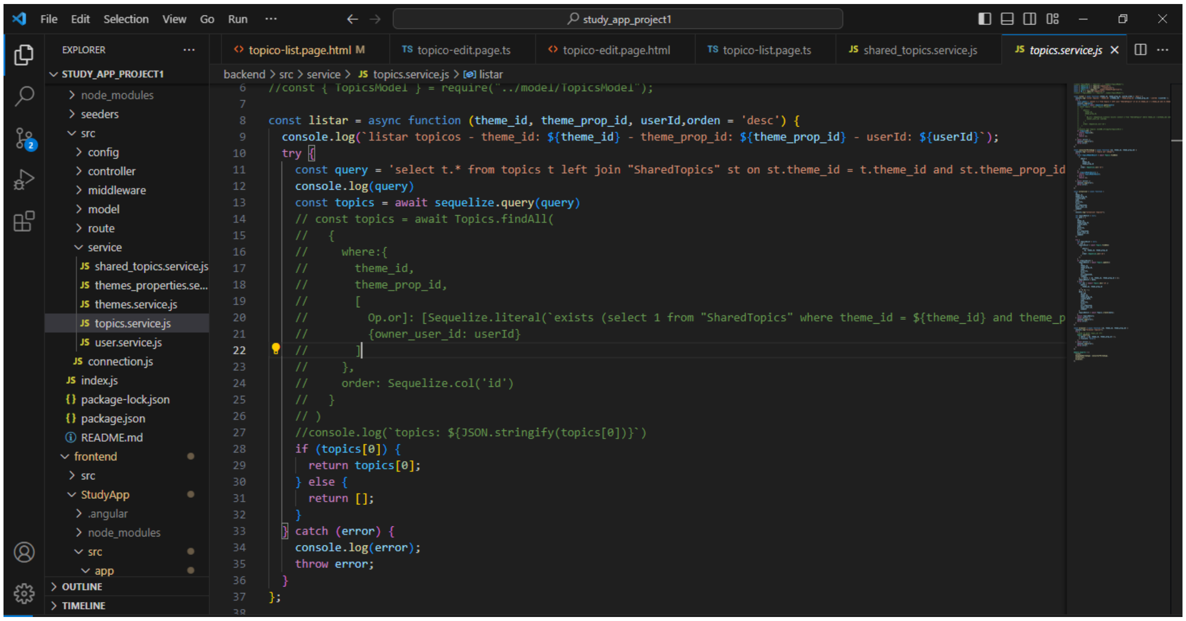
ORDER TOPIC



```
  ngOnInit() {
    this.theme_id = this.activatedRoute.snapshot.paramMap.get('theme_id') as string;
    this.theme_prop_id = this.activatedRoute.snapshot.paramMap.get('theme_prop_id') as string;
    this.userId = localStorage.getItem("user_id") ??'';
    //this.getTopics();
  }

  ordenAsc(){
    this.orden = 'asc'
    this.getTopics('asc')
  }

  ordenDesc(){
    this.orden = 'desc'
    this.getTopics('desc')
  }


  async confirmDelete(id: string, theme_id: string, theme_prop_id: string) {
    const alert = await this.alertController.create({
      header: 'Mensaje',
      message: 'Desea eliminar el Topico?',
      buttons: [
        {
          text: 'Confirmar ',
          handler: () => {
            this.deleteTheme(id,theme_id,theme_prop_id);
          },
        },
        {
          text: 'Cancelar',
          handler: () => {
```

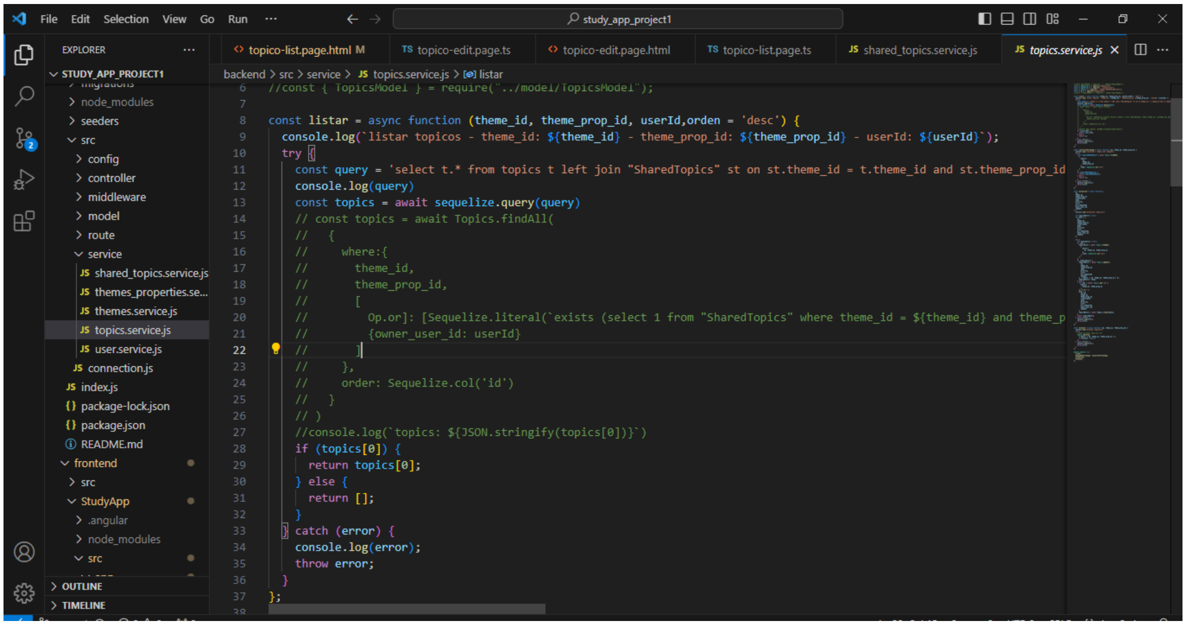


```
          },
        },
      ],
    });
    await alert.present();
  }

  getTopics(orden: string) {
    let headers = this.getHeader();
    axios
      .get('http://localhost:3000/topics/list/'+this.theme_id+'/'+this.theme_prop_id+'/'+orden,headers)
      .then((result) => {
        if (result.data.success == true) {
          if(result.data.topicos && result.data.topicos[0]){
            //console.log(`result.data.topicos: ${JSON.stringify(result.data.topicos)}`)
            this.topicos = result.data.topicos;
          }
        } else {
          console.log(result.data.error);
        }
      })
      .catch((error) => {
        console.log(error.message);
      });
  }

  getPropName(){
    let headers = this.getHeader();
      axios
        .get('http://localhost:3000/themeproperties/buscarPorCodigo/' + this.theme_id + '/' + this.theme_prop_id,headers
        .then((result) => {
          if (result.data.success == true) {
            //console.log(`result.data: ${JSON.stringify(result.data)}`)
```

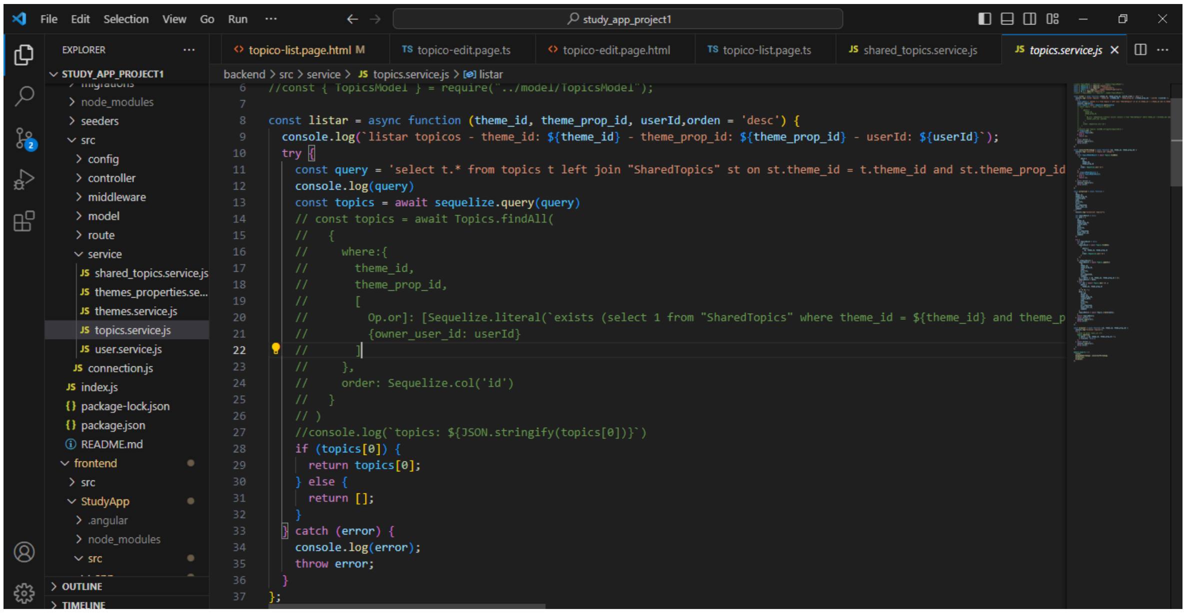


```javascript
//const { TopicsModel } = require("../model/TopicsModel");

const listar = async function (theme_id, theme_prop_id, userId,orden = 'desc') {
  console.log(`listar topicos - theme_id: ${theme_id} - theme_prop_id: ${theme_prop_id} - userId: ${userId}`);
  try {
    const query = 'select t.* from topics t left join "SharedTopics" st on st.theme_id = t.theme_id and st.theme_prop_id
    console.log(query)
    const topics = await sequelize.query(query)
    // const topics = await Topics.findAll(
    //   {
    //     where:{
    //       theme_id,
    //       theme_prop_id,
    //       [
    //         Op.or]: [Sequelize.literal(`exists (select 1 from "SharedTopics" where theme_id = ${theme_id} and theme_p
    //         {owner_user_id: userId}
    //       ]
    //     },
    //     order: Sequelize.col('id')
    //   }
    // )
    //console.log(`topics: ${JSON.stringify(topics[0])}`)
    if (topics[0]) {
      return topics[0];
    } else {
      return [];
    }
  } catch (error) {
    console.log(error);
    throw error;
  }
};
```

PRIORITY TOPIC



```javascript
//const { TopicsModel } = require("../model/TopicsModel");

const listar = async function (theme_id, theme_prop_id, userId,orden = 'desc') {
  console.log(`listar topicos - theme_id: ${theme_id} - theme_prop_id: ${theme_prop_id} - userId: ${userId}`);
  try {
    const query = 'select t.* from topics t left join "SharedTopics" st on st.theme_id = t.theme_id and st.theme_prop_id
    console.log(query)
    const topics = await sequelize.query(query)
    // const topics = await Topics.findAll(
    //   {
    //     where:{
    //       theme_id,
    //       theme_prop_id,
    //       [
    //         Op.or]: [Sequelize.literal(`exists (select 1 from "SharedTopics" where theme_id = ${theme_id} and theme_p
    //         {owner_user_id: userId}
    //       ]
    //     },
    //     order: Sequelize.col('id')
    //   }
    // )
    //console.log(`topics: ${JSON.stringify(topics[0])}`)
    if (topics[0]) {
      return topics[0];
    } else {
      return [];
    }
  } catch (error) {
    console.log(error);
    throw error;
  }
};
```



```
//const { TopicsModel } = require("../model/TopicsModel");

const listar = async function (theme_id, theme_prop_id, userId,orden = 'desc') {
  console.log(`listar topicos - theme_id: ${theme_id} - theme_prop_id: ${theme_prop_id} - userId: ${userId}`);
  try {
    const query = 'select t.* from topics t left join "SharedTopics" st on st.theme_id = t.theme_id and st.theme_prop_id
    console.log(query)
    const topics = await sequelize.query(query)
    // const topics = await Topics.findAll(
    //   {
    //     where:{
    //       theme_id,
    //       theme_prop_id,
    //       [
    //         Op.or]: [Sequelize.literal(`exists (select 1 from "SharedTopics" where theme_id = ${theme_id} and theme_p
    //         {owner_user_id: userId}
    //       ]
    //     },
    //     order: Sequelize.col('id')
    //   }
    // )
    //console.log(`topics: ${JSON.stringify(topics[0])}`)
    if (topics[0]) {
      return topics[0];
    } else {
      return [];
    }
  } catch (error) {
    console.log(error);
    throw error;
  }
};
```