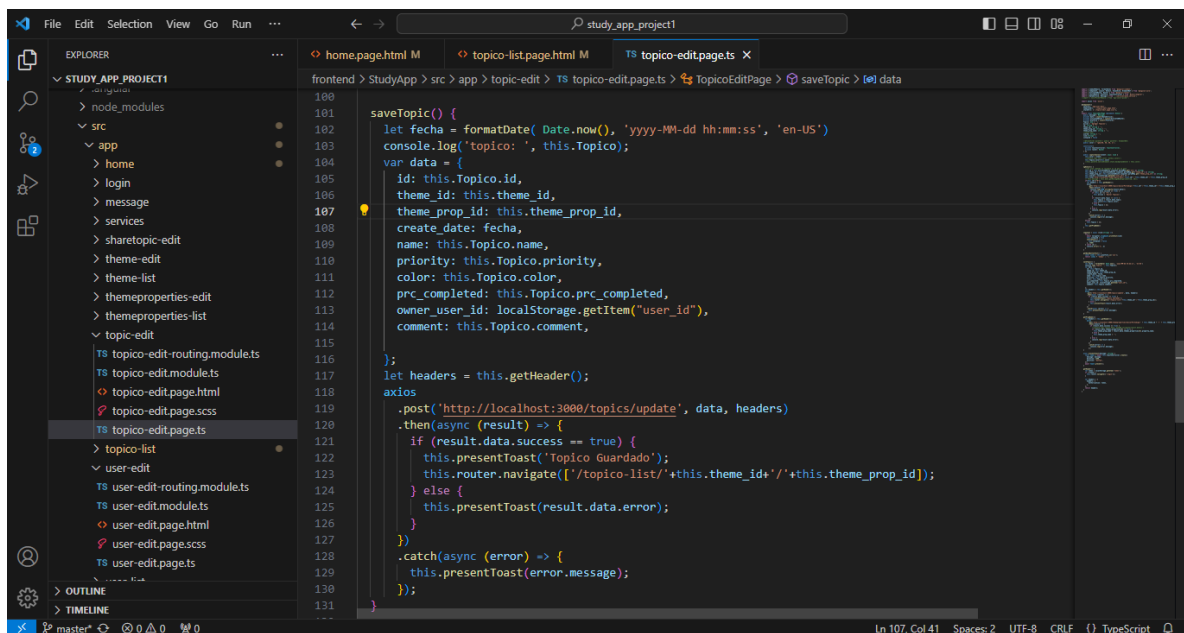


Robert Elias Leguizamon Gonzalez

5939456

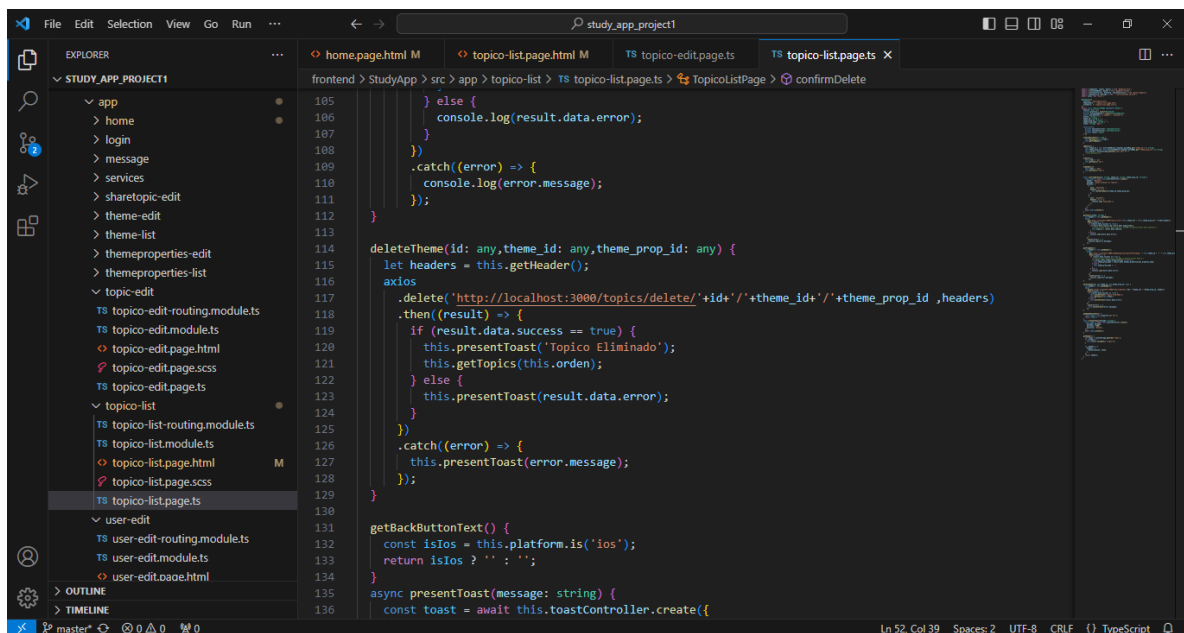
Examen Final

TOPICOS



The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying the project structure. The main editor window shows the `topic-edit.page.ts` file. The `saveTopic()` function is highlighted, showing the following code:

```
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);  
    });  
}
```



The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying the project structure. The main editor window shows the `topic-list.page.ts` file. The `deleteTheme()` function and `getBackButtonText()` method are highlighted, showing the following code:

```
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({  
    message: message,  
    duration: 3000,  
    position: 'top',  
    color: 'info',  
  });  
  toast.present();  
}
```

The screenshot shows the VS Code editor with the Explorer sidebar on the left displaying the project structure. The main editor window shows the file `topics.service.js` with the following code:

```
90
91
92 try {
93   let topicsExist = null;
94   if (id) {
95     topicsExist = await Topics.findOne(
96       {
97         where: {
98           id, theme_id, theme_prop_id
99         },
100         order: Sequelize.col('id')
101       }
102     );
103   }
104   if (topicsExist) {
105     topicsReturn = await Topics.update({
106       id,
107       theme_id,
108       theme_prop_id,
109       name,
110       priority,
111       color,
112       prc_completed,
113       comment
114     }, { where: { id, theme_id, theme_prop_id } });
115     topicsReturn = data;
116   } else {
117     let id2 = (await Topics.max('id', {
118       where: {
119         theme_id, theme_prop_id
120       }
121     ))) ?? 0 + 1
122     data = {
```

The screenshot shows the VS Code editor with the Explorer sidebar on the left displaying the project structure. The main editor window shows the file `topics.service.js` with the following code:

```
111 prc_completed,
112 comment
113 }, { where: { id, theme_id, theme_prop_id } });
114 topicsReturn = data;
115 } else {
116   let id2 = (await Topics.max('id', {
117     where: {
118       theme_id, theme_prop_id
119     }
120   ))) ?? 0 + 1
121   data = {
122     id: id2,
123     theme_id,
124     theme_prop_id,
125     create_date,
126     name,
127     priority,
128     color,
129     prc_completed,
130     owner_user_id,
131     comment
132   }
133   topicsReturn = await Topics.create(data);
134 }
135 return topicsReturn;
136 } catch (error) {
137   console.log(error);
138   throw error;
139 }
140 };
141
142 const eliminar = async function (id, theme_id, theme_prop_id) {
```

THEMES PROPERTIES

The screenshot shows the VS Code editor with the file explorer on the left displaying the project structure. The main editor window shows the `themeproperties-edit.page.ts` file. The `saveThemeProperty` method is highlighted, showing a call to `axios.post` to update a theme property. The breadcrumb navigation at the top indicates the path: `StudyApp > src > app > themeproperties-edit > TS themeproperties-edit.page.ts > ThemePropertiesEditPage > saveThemeProperty > then0 callback`.

```
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) {
```

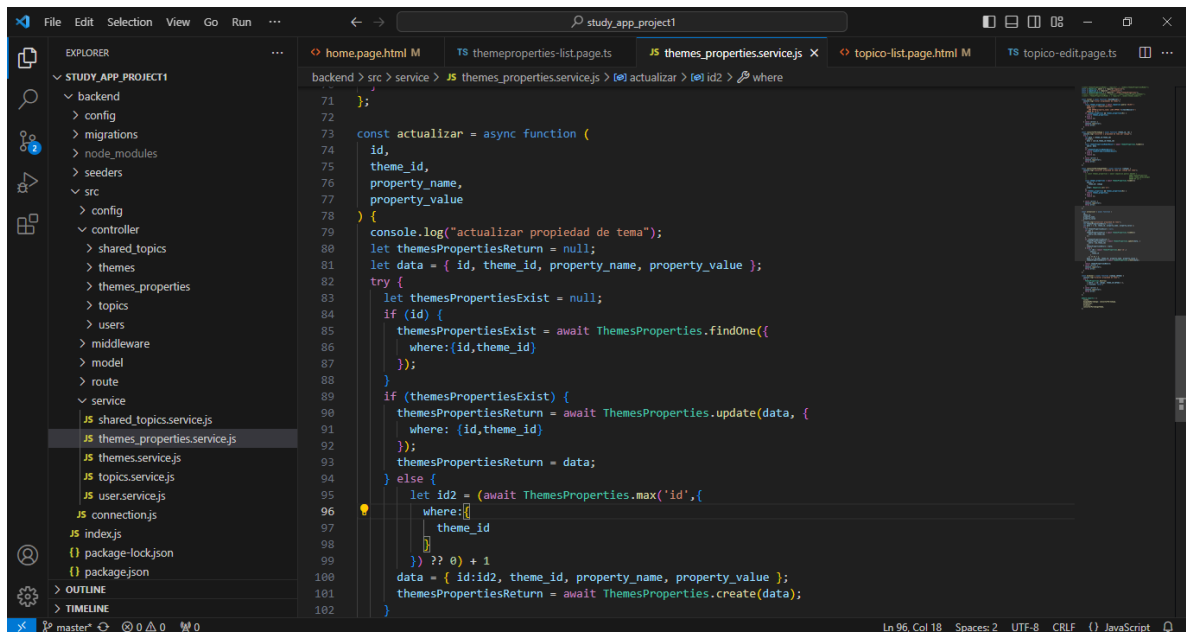
The screenshot shows the VS Code editor with the file explorer on the left. The main editor window shows the `themeproperties-list.page.ts` file. The `deleteThemeProperties` method is highlighted, showing a call to `axios.delete` to remove a theme property. The breadcrumb navigation at the top indicates the path: `StudyApp > src > app > themeproperties-list > TS themeproperties-list.page.ts > ...`.

```
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);
});
}

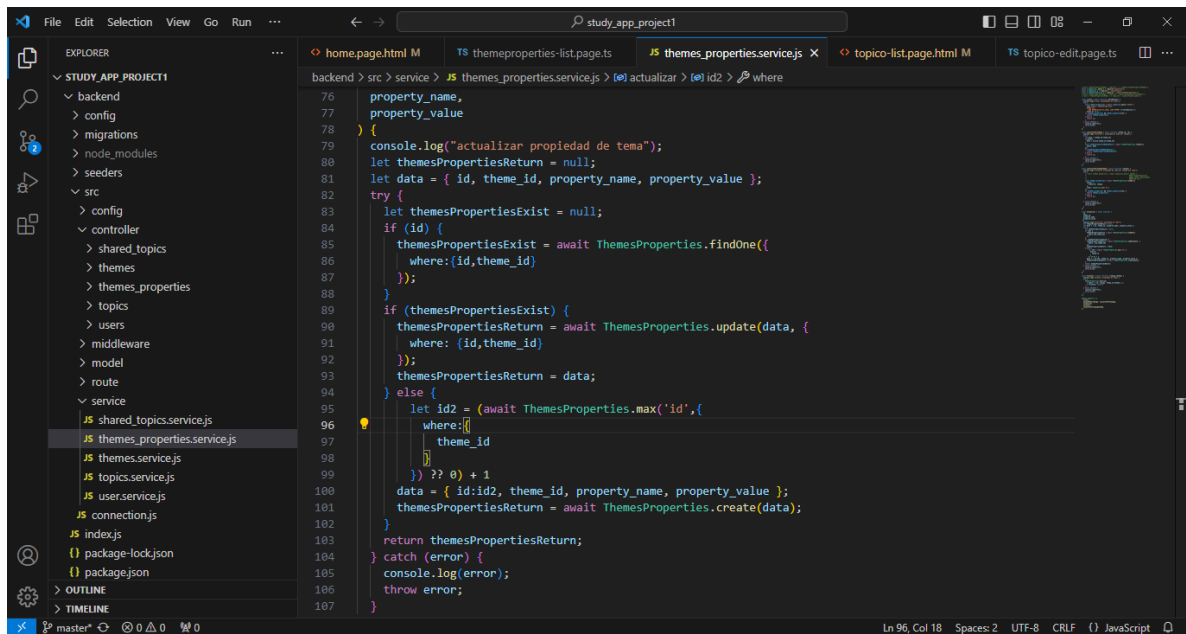
deleteThemeProperties(id: any) {
  let headers = this.getHeader();
  axios
    .delete('http://localhost:3000/themeproperties/delete/' + id, headers)
    .then((result) => {
      if (result.data.success == true) {
        this.presentToast('Tema Eliminado');
        this.getThemeProperties();
      } 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({
```



```
71;
72;
73const actualizar = async function (
74  id,
75  theme_id,
76  property_name,
77  property_value
78) {
79  console.log("actualizar propiedad de tema");
80  let themesPropertiesReturn = null;
81  let data = { id, theme_id, property_name, property_value };
82  try {
83    let themesPropertiesExist = null;
84    if (id) {
85      themesPropertiesExist = await ThemesProperties.findOne({
86        where: { id, theme_id }
87      });
88    }
89    if (themesPropertiesExist) {
90      themesPropertiesReturn = await ThemesProperties.update(data, {
91        where: { id, theme_id }
92      });
93      themesPropertiesReturn = data;
94    } else {
95      let id2 = (await ThemesProperties.max('id', {
96        where: {
97          theme_id
98        }
99      }) ?? 0) + 1
100      data = { id: id2, theme_id, property_name, property_value };
101      themesPropertiesReturn = await ThemesProperties.create(data);
102    }
103  } catch (error) {
104    console.log(error);
105    throw error;
106  }
107  return themesPropertiesReturn;
108}
```



```
76  property_name,
77  property_value
78) {
79  console.log("actualizar propiedad de tema");
80  let themesPropertiesReturn = null;
81  let data = { id, theme_id, property_name, property_value };
82  try {
83    let themesPropertiesExist = null;
84    if (id) {
85      themesPropertiesExist = await ThemesProperties.findOne({
86        where: { id, theme_id }
87      });
88    }
89    if (themesPropertiesExist) {
90      themesPropertiesReturn = await ThemesProperties.update(data, {
91        where: { id, theme_id }
92      });
93      themesPropertiesReturn = data;
94    } else {
95      let id2 = (await ThemesProperties.max('id', {
96        where: {
97          theme_id
98        }
99      }) ?? 0) + 1
100      data = { id: id2, theme_id, property_name, property_value };
101      themesPropertiesReturn = await ThemesProperties.create(data);
102    }
103    return themesPropertiesReturn;
104  } catch (error) {
105    console.log(error);
106    throw error;
107  }
108}
```

SHARED TOPICOS

```
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106

frontend > StudyApp > src > app > sharetopico-edit > TS sharetopico-edit.page.ts > ShareTopicoEditPage

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);
    });
}
```

```
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119

backend > src > service > JS shared_topics.service.js > ...

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

```

The screenshot shows the VS Code editor with the Explorer sidebar on the left displaying the project structure. The main editor window shows the file `shared_topics.service.js` with the following code:

```
backend > src > service > JS shared_topics.service.js > actualizar
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
};

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;
}
```

The screenshot shows the VS Code editor with the Explorer sidebar on the left displaying the project structure. The main editor window shows the file `themes_properties.service.js` with the following code:

```
JS themes_properties.service.js > actualizar > id2 > where
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
};

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,
};
```

```
68 },
69 ],
70 });
71 await alert.present();
72 }
73
74 getTopics(ordem: string) {
75   let headers = this.getHeader();
76   axios
77     .get('http://localhost:3000/topics/list/' + this.theme_id + '/' + this.theme_prop_id + '/' + ordem, headers)
78     .then((result) => {
79       if (result.data.success == true) {
80         if (result.data.topicos && result.data.topicos[0]) {
81           //console.log('result.data.topicos: ' + JSON.stringify(result.data.topicos))
82           this.topicos = result.data.topicos;
83         }
84       } else {
85         console.log(result.data.error);
86       }
87     })
88     .catch((error) => {
89       console.log(error.message);
90     });
91 }
92
93 getPropName(){
94   let headers = this.getHeader();
95   axios
96     .get('http://localhost:3000/themesproperties/buscarPorCodigo/' + this.theme_id + '/' + this.theme_pro
```

VER TOPICOS COMPARTIDOS

```
6 //const { TopicsModel } = require('../models/topicsModel');
7
8
9 const listar = async function (theme_id, theme_prop_id, userId, ordem = 'desc') {
10   console.log('listar topicos - theme_id: ' + theme_id + ' - theme_prop_id: ' + theme_prop_id + ' - userId: ' + userId);
11   try {
12     const query = 'select t.* from topics t left join "SharedTopics" st on st.theme_id = t.theme_id and st.theme_prop_id = t.theme_prop_id';
13     console.log(query);
14     const topics = await sequelize.query(query)
15     // const topics = await Topics.findAll({
16     //   where: {
17     //     theme_id,
18     //     theme_prop_id,
19     //     [Op.or]: [Sequelize.literal('exists (select 1 from "SharedTopics" where theme_id = ' + theme_id + ' and theme_p' + theme_prop_id + ' and owner_user_id = ' + userId + ')')]
20     //   }
21     // }, {
22     //   order: Sequelize.col('id')
23     // });
24     //console.log('topics: ' + JSON.stringify(topics[0]))
25     if (topics[0]) {
26       return topics[0];
27     } else {
28       return [];
29     }
30   } catch (error) {
31     console.log(error);
32     throw error;
33   }
34 };
```

ORDER TOPIC

The screenshot shows the VS Code editor with the file explorer on the left displaying the project structure. The main editor window shows the 'topico-list.page.ts' file. The 'confirmDelete' method is highlighted, which is an asynchronous function that takes 'id', 'theme_id', and 'theme_prop_id' as parameters. It uses 'this.alertController.create()' to show a confirmation dialog with the message 'Desea eliminar el Topico?'. The dialog has two buttons: 'Confirmar' and 'Cancelar'. The 'Confirmar' button's handler calls 'this.deleteTheme(id, theme_id, theme_prop_id)'.

```
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',
    },
  ],
});
```

The screenshot shows the VS Code editor with the file explorer on the left. The main editor window shows the 'topico-list.page.ts' file. The 'getTopics' method is highlighted, which is a synchronous function that takes 'orden' as a parameter. It uses 'this.getHeader()' to get the headers and 'axios.get()' to fetch the topics from the API. The response is checked for success, and if so, the 'this.topicos' property is updated with the result. If there is an error, it is logged to the console. The 'getPropName' method is also shown, which is a synchronous function that takes no parameters and uses 'this.getHeader()' to get the headers and 'axios.get()' to fetch the theme properties from the API. The response is checked for success, and if so, the 'this.theme_prop_id' property is updated with the result.

```
},
},
});
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))
      }
    })
    .catch((error) => {
      console.log(error.message);
    });
}
```



```
back&gt; src > service > JS topics.service.js > listar
6 //const { TopicsModel } = require( '../model/TopicsModel' );
7
8 const listar = async function ( theme_id, theme_prop_id, userId,orden = 'desc' ) {
9   console.log( 'listar topicos - theme_id: ${theme_id} - theme_prop_id: ${theme_prop_id} - userId: ${userId}' );
10  try {
11    const query = 'select t.* from topics t left join "SharedTopics" st on st.theme_id = t.theme_id and st.theme_prop_id
12    console.log(query)
13    const topics = await sequelize.query(query)
14    // const topics = await Topics.findAll(
15    //   {
16    //     where:{
17    //       theme_id,
18    //       theme_prop_id,
19    //       [
20    //         Op.or: [Sequelize.literal("exists (select 1 from "SharedTopics" where theme_id = ${theme_id} and theme_p
21    //         {owner_user_id: userId}
22    //       ]
23    //     },
24    //     order: Sequelize.col('id')
25    //   )
26    //   )
27    //console.log("topics: ${JSON.stringify(topics[0])}")
28    if (topics[0]) {
29      return topics[0];
30    } else {
31      return [];
32    }
33  } catch (error) {
34    console.log(error);
35    throw error;
36  }
37 };
```

PRIORITY TOPIC

```
back&gt; src > service > JS topics.service.js > listar
6 //const { TopicsModel } = require( '../model/TopicsModel' );
7
8 const listar = async function ( theme_id, theme_prop_id, userId,orden = 'desc' ) {
9   console.log( 'listar topicos - theme_id: ${theme_id} - theme_prop_id: ${theme_prop_id} - userId: ${userId}' );
10  try {
11    const query = 'select t.* from topics t left join "SharedTopics" st on st.theme_id = t.theme_id and st.theme_prop_id
12    console.log(query)
13    const topics = await sequelize.query(query)
14    // const topics = await Topics.findAll(
15    //   {
16    //     where:{
17    //       theme_id,
18    //       theme_prop_id,
19    //       [
20    //         Op.or: [Sequelize.literal("exists (select 1 from "SharedTopics" where theme_id = ${theme_id} and theme_p
21    //         {owner_user_id: userId}
22    //       ]
23    //     },
24    //     order: Sequelize.col('id')
25    //   )
26    //   )
27    //console.log("topics: ${JSON.stringify(topics[0])}")
28    if (topics[0]) {
29      return topics[0];
30    } else {
31      return [];
32    }
33  } catch (error) {
34    console.log(error);
35    throw error;
36  }
37 };
```

The screenshot shows a Visual Studio Code editor window with a project named 'study_app_project1'. The Explorer sidebar on the left displays the project structure, including a 'src' directory with files like 'shared_topics.service.js', 'themes_properties.se...', 'topics.service.js', and 'user.service.js'. The main editor area shows the 'topics.service.js' file, which contains a REST client for the 'listar' endpoint. The code defines an async function 'listar' that takes 'theme_id', 'theme_prop_id', 'userId', and 'orden' as parameters. It constructs a SQL query to select topics from a 'topics' table, joined with a 'SharedTopics' table. The query is logged to the console, and the results are returned as a JSON array. The code also includes error handling and a console log for the topics.

```
backend > src > service > JS topics.service.js > listar
6 //const { TopicsModel } = require( '../model/TopicsModel' );
7
8 const listar = async function (theme_id, theme_prop_id, userId,orden = 'desc') {
9   console.log('listar topics - theme_id: ${theme_id} - theme_prop_id: ${theme_prop_id} - userId: ${userId}');
10  try {
11    const query = 'select t.* from topics t left join "SharedTopics" st on st.theme_id = t.theme_id and st.theme_prop_id = t.theme_prop_id';
12    console.log(query);
13    const topics = await sequelize.query(query);
14    // const topics = await Topics.findAll(
15    //   {
16    //     where:{
17    //       theme_id,
18    //       theme_prop_id,
19    //       [Op.or]: [Sequelize.literal('exists (select 1 from "SharedTopics" where theme_id = ${theme_id} and theme_prop_id = ${theme_prop_id} and owner_user_id = ${userId})')]
20    //     }
21    //   },
22    //   {
23    //     order: Sequelize.col('id')
24    //   }
25    // );
26    //console.log('topics: ${JSON.stringify(topics[0])}')
27    if (topics[0]) {
28      return topics[0];
29    } else {
30      return [];
31    }
32  } catch (error) {
33    console.log(error);
34    throw error;
35  }
36 }
37
```