Programação III

Home Managing WebApp

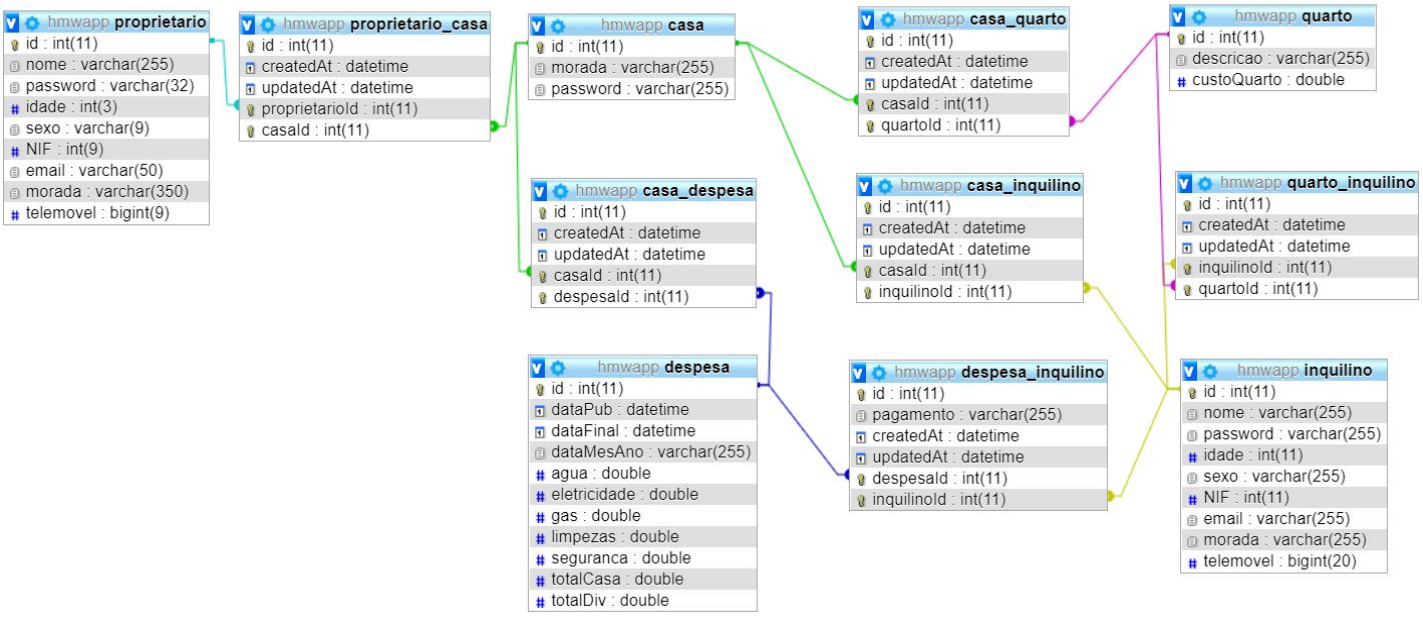
Ema Guedes, nº 11280

2º ano de Multimédia
Instituto Superior Miguel Torga

Tema

A aplicação tem como função ajudar os proprietários de residências partilhadas na gestão das despesas. Os inquilinos dessas residências, também podem aceder ao site, estando sempre notificados das despesas que têm de pagar ou verem o histórico de todas as despesas associadas a essa residência tal como uma lista de nomes de todas as pessoas que lá habitam. Se quiserem, têm acesso às informações do administrador para que o consigam contactar ou fazer uma transferência bancária.

Base de dados



Os proprietários têm casas, as casas têm despesas e quartos, os quartos têm inquilinos. Os inquilinos também têm despesas.

Design

Escolhemos diferentes azuis, pois esta cor simboliza confiança, cooperatividade, competência e alta qualidade. Queremos que quem utilize a nossa aplicação, se sinta seguro e que acredite na nossa competência.
O logo da aplicação, representa o equilíbrio econômico na casa. Sendo que a balança representa justiça, dando assim a entender, que a aplicação é de confiança.

Código

API

Foram usados todos os métodos aprendidos em aula: GET, POST, PUT e DELETE. O GET para buscar valores à base de dados(ex: buscar informação de um inquilino), o POST para inserir valores à base de dados(ex: para criar um inquilino); PUT para atualizar valores na base de dados(ex: para mudar a idade de um inquilino) e DELETE para eliminar valores da base de dados(ex: para eliminar um quarto).

src > config > JS database.js > ...

```
1   const sequelize = require('sequelize');
2   const ligacao = new sequelize('hmwapp', 'root', '', {
3       host: 'localhost',
4       dialect: 'mysql'
5   });
6   module.exports = ligacao;
```

Este código é o que permite ligar a base de dados à aplicação. O « 'hmwapp' » é o nome da base de dados.
Os ficheiros estão organizados de acordo com as normas dadas pelo professor.

```
const nInquilinos = await CasaInquilino.findAll({
  where: { casaId: 1 },  //mudar a casaId
  attributes: ['casaId', [sequelize.fn('count', sequelize.col('inquilinoId')), 'nInquilinos'],],
  group: ['casaId'],
  raw:true})
  .then(function (nInquilinos) {
    console.log(nInquilinos);
    return nInquilinos;
  })
  .catch((error) => {
    console.log(error);
  });
```

Este é o código para contar o número de inquilinos, neste caso, da casa 1. O "where" tem de ser melhorado.

```
const totalCasa = await DespesaInquilino.findAll({
  attributes: {
    include: [
      [sequelize.literal(`(SELECT agua+eletricidade+gas+limpezas+seguranca FROM despesa)`),
        'totalDespesa'
      ]
    ]
  },
raw:true})
  .then(function (totalCasa) {
    console.log(totalCasa);
    return totalCasa;
  })
  .catch((error) => {
    console.log(error);
  });
```

Este código, faz a soma da despesa “agua”, “eletricidade”,...

```
let totalDiv = parseFloat(totalCasa[0]["totalDespesa"])/parseInt(nInquilinos[0]["nInquilinos"]);
res.json({
  success: true,
  nInquilinos: nInquilinos,
  despesa: despesa,
  totalCasa: totalCasa,  // ---> não descobri o motivo, mas ele não está a receber os valores. apa
  totalDiv: totalDiv,
});
```

Este código divide o total das despesas pelo número de inquilinos.

```
},
pagamento: {
    type: sequelize.STRING,
    defaultValue: "Por pagar"
},
```

Este código cria o defaulValue “Por Pagar” para a string “pagamento”.

```
},
dataPub: {
    type: sequelize.DATE,
    defaultValue: sequelize.NOW
},
```

Este código é uma das maneiras possíveis, e que me pareceu mais correcta, para guardar a data e o tempo exacto em que a despesa foi criada/publicada.

Android Studio

Manifest

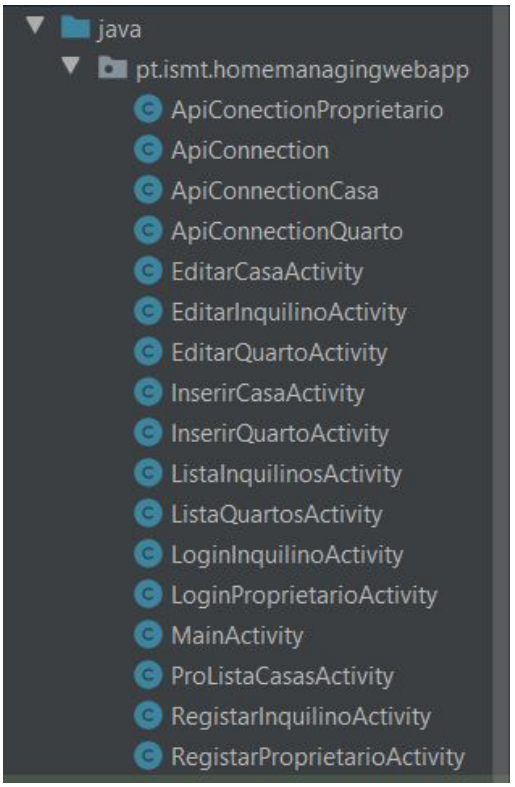
```
android:icon="@mipmap/ic_logo"
android:label="Home Managing WebApp"
android:roundIcon="@mipmap/ic_logo_round"
```

Usei estas linhas para definir o nome da minha aplicação e o logo da mesma.

```
<uses-permission android:name="android.permission.INTERNET" />
```

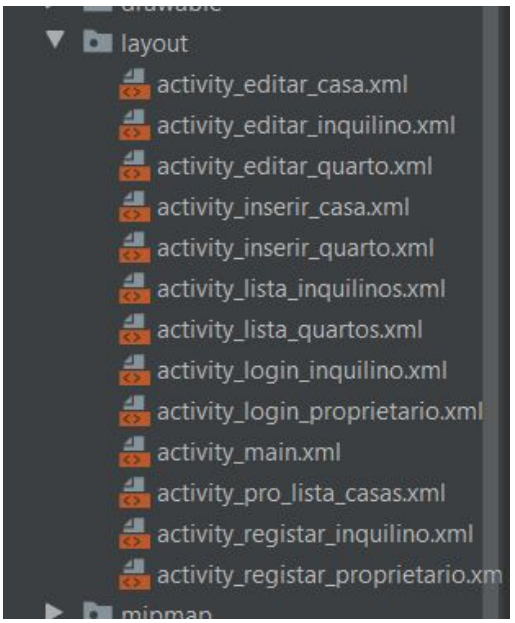
```
android:usesCleartextTraffic="true">
```

Estas duas linhas são necessárias para a aplicação funcionar.



Java

Na pasta pt.ismt.homemanagingwebapp, criei todos os documentos java para que fosse possível estabelecer a ligação da API com o Android Studio, possibilitando inserir(POST), buscar(GET) ou editar(PUT) valores da base de dados.



Layout

Na pasta layout ,dentro da pasta res, criei o design de todas as páginas da aplicação.

Values

Na pasta strings.xml, criei todas as strings que aparecem na aplicação.
Na pasta colors.xml, acrescentei duas cores personalizadas que eu escolhi para o site.
Os 3 documentos xml que começam por ic_(...) foram gerados automaticamente quando criei a imagem do logo, a imagem com o ícone do proprietário e o ícone do inquilino.

Aspetos a melhorar

A aplicação Android tem de ser melhor estruturada e devidamente completada. O facto da lista inquilinos não aparecer, ser corrigido. Os *login* dos utilizadores precisam de ser feitos, para assim também aparecerem as informações do inquilino ou proprietário em questão. Apesar disso estão presentes dois botões, um com “ver perfil” e outro com “Atualizar perfil”, para ficar claro que o utilizador tem essas opções. A parte do inquilino deveria de ser completada. Não ter de um java para cada ApiConnection.
Conseguir com que o código relacionado às contas das despesas funcione, inserindo os resultados, das mesmas à base de dados na respectiva coluna da tabela. Fazer o código para enviar um pedido para uma casa e a confirmação da mesma para a pessoa ser um inquilino dessa residência. Aparecer o estado do pagamento “Por pagar”, e puder atualizá-lo para “Pago”.

Progressão

A API não foi concluída devido ao tempo e complexidade. O cálculo das despesas foi uma etapa muito complicada, da qual necessitei de ajuda do professor. Ainda assim, modificando posteriormente o código das despesas, o mesmo deixou de funcionar correctamente. Diria que o cálculo das despesas foi a etapa mais difícil e que levou mais tempo a ser realizada.
Relativamente ao Android Studio, tive muitas dificuldades devido ao intervalo de tempo que me foi disponibilizado para entregar o trabalho. Como estive muito focada em aperfeiçoar e completar a API e o meu Android Studio não estava a funcionar, não aprendi muito sobre o mesmo. Esta última etapa, desafiou as minhas capacidades de aprender, executar e concentrar. Apesar da aplicação Android estar incompleta, estou satisfeita com o resultado. Gostaria de poder ter entregue um trabalho mais bem executado, mas tendo em conta o tempo, decidi fazer o que estava dentro dos meus limites e um pouco além.

Conclusão

O trabalho ficou incompleto, mas sinto-me satisfeita com o mesmo, tendo em conta as circunstâncias. No final, consegui 2 listas(GET): casas e quartos; 4 POST: registar inquilino, registar proprietário, adicionar casa e adicionar quarto; 2 PUT: casa e quarto.
As aulas foram esclarecedoras, e mesmo se um aluno tivesse dúvidas, o professor ajudaria-o consoante a disponibilidade do mesmo. Um dos factores mais importantes que me auxiliou muito nesta disciplina, foram as fichas/projectos disponibilizados disponibilizadas no moodle pelo professor, facilitando a aprendizagem dos temas.

Bibliografia

https://sequelize.org/v3/docs/models-definition/ - para saber as saber as definições dos models. Depois aplicadas nos ficheiros da pasta models.
https://stackoverflow.com/questions/29652538/sequelize-js-timestamp-not-datetime e
https://sequelize.org/v5/manual/models-definition.html - para receber o tempo e data real em que a despesa foi criada, sendo representada na coluna dataPub da base de dados.
https://sequelize.org/master/manual/model-basics.html - para saber como fazer um defaultValue para string. Usado na coluna pagamento.
https://sequelize.readthedocs.io/en/latest/docs/associations/ - para saber as relacões e como as implementar.