Universidade Europeia

IADE

QuickWork

U'c: Projeto de Desenvolvimento Móvel

Professor: Pedro Rosa

Turma 1

https://github.com/fransantos1/QUICKWORK

Francisco Santos - 20211206

Gonçalo Santos - 50037145

Gustavo Farinha - 20211115

Faculdade de Design, Tecnologia e Comunicação
Universidade Europeia

Descrição da aplicação

QuickWork será uma aplicação que permitirá ajudar pessoas que precisam de assistência em tarefas domésticas, ou que não têm capacidades financeiras para comprar ferramentas profissionais, dando-lhes a hipótese de pedir ajuda no serviço, e ter uma outra pessoa que tenha as qualificações/ferramentas/tempo para ir auxiliar.

Uma pessoa que vá auxiliar também poderá pedir ajuda na aplicação, necessitando assim um único tipo de conta, para auxiliar ou pedir ajuda.

Motivação do trabalho a realizar

Esta aplicação tem como objetivo ajudar pessoas que não têm a capacidade de fazer certos trabalhos, ou porque estão sozinhas, não têm capacidades físicas ou não sentem que vale a pena comprar ferramentas caras para um trabalho que demore uma tarde.

Público-Alvo

Como foi referido no ponto anterior, o público-alvo desta aplicação seria quem estivesse em desvantagem física, ou com pouca capacidade financeira. Este serviço também servirá para os jovens adultos (18-25 anos) que não tenham disponibilidade para um "full time job", terem maneira de ganhar dinheiro com tarefas simples nos seus tempos livres.

Aplicações Semelhantes

QuickWork relaciona-se melhor com Fiverr, mas em vez de ser só com tarefas online seria presencialmente e seria o cliente a meter o anúncio em vez do vendedor. Outras aplicações que encontrei que seriam parecidas seriam:

Gigwalk - que seria a aplicação mais parecida à QuickWork, mas, não consegui entrar na aplicação.

TaskRabbit - esta aplicação tem um login diferente para trabalhador e cliente, com a nossa aplicação, estamos a pensar que um trabalhador também pode ser cliente e vice-versa.

Olx - esta aplicação tem uma divisão de temas para a venda e compra de objetos, na nossa aplicação vamos dividir os trabalhos em temas e a pessoa escolhe o tema onde se quer integrar.

Guiões

Utilização do nosso objeto "Core" na aplicação:

O objeto core da nossa aplicação seria a utilização do mapa, que seria onde o utilizador poderá visualizar todos os trabalhos disponíveis e escolher qual pretende fazer, ou onde poderá também publicar um trabalho.

Criar uma conta:

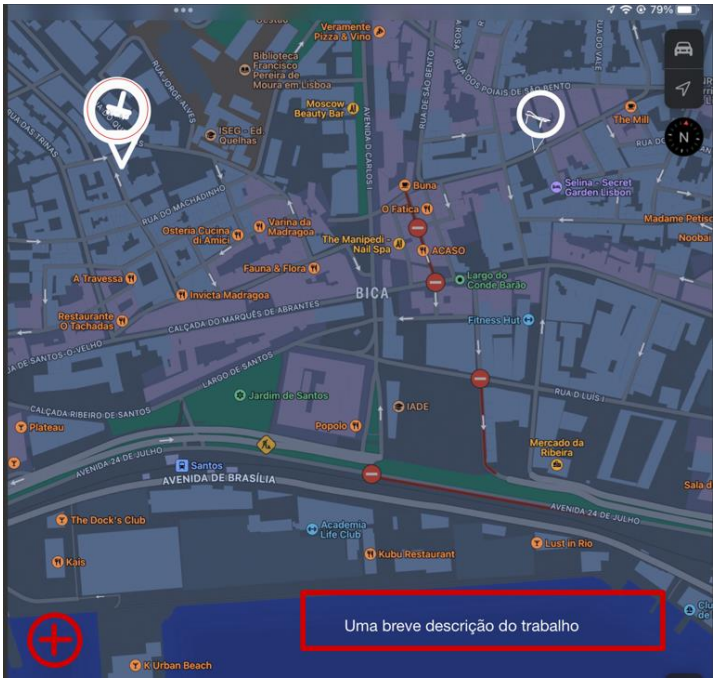
Quando um utilizador abre a aplicação pela primeira é-lhe mostrada uma página de login, onde, se o utilizador não tiver conta ainda, pode carregar num botão que diz "criar conta".

Depois do utilizador carregar no botão é pedido para inserir um email, nome de usuário, password e confirmar a password.

Na próxima página também é perguntado ao utilizador quais são os tipos de trabalho em que se encontra mais confortável.

Após o utilizador inserir os seus dados carrega no botão de criar conta e a conta é criada.

Criar tarefa:



Quando se abre a aplicação, a primeira janela será o mapa, aí o utilizador, em baixo terá um botão com um sinal de '+', ao clicar nesse botão poderá escolher, "criar trabalho", insere os dados do trabalho:

- Qual é o trabalho, localização
- estimativa de duração
- Escolher o tema do trabalho
- Preço que pretende pagar por esse trabalho

A seguir de inserir esses dados o utilizador confirma que quer criar a tarefa, e a tarefa é criada.

Quando a tarefa é realizada, o "cliente" pode classificar o "trabalhador" e deixar algum comentário, se pretender, o vice-versa também ocorre.

Escolher tarefa:

No mapa, aparecerá, as tarefas disponíveis, aí poderá o utilizador escolher a tarefa que pretende fazer, quando escolher a tarefa, é preciso apenas carrega na bolha que aparece no mapa, vê o que é que o cliente pretende, e terá acesso também ao rating desse cliente e os comentários, depois o fale com o cliente e vai fazer a tarefa. A seguir quando a tarefa é acabada o trabalhador é pago e pode deixar rating e comentários no cliente, e o vice-versa também ocorre.

Perfil:

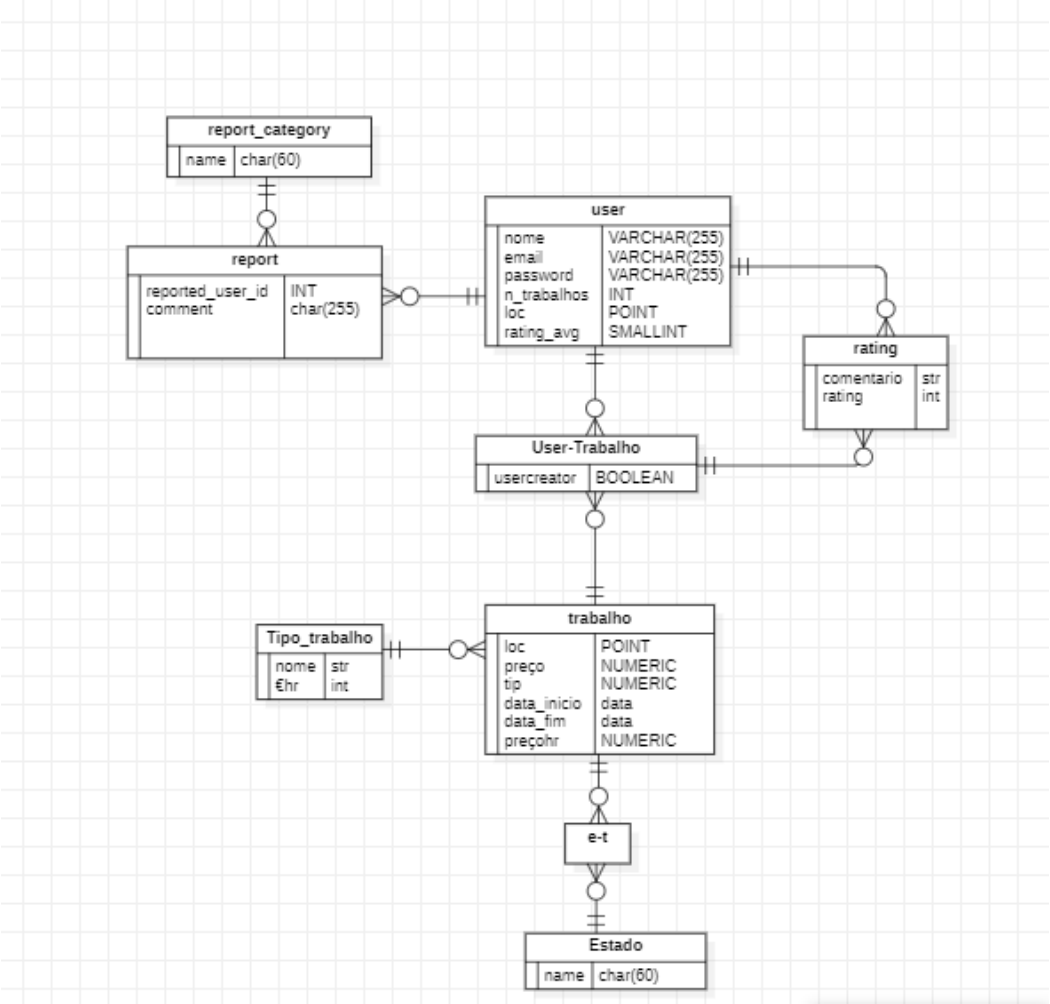
O perfil vai conter uma série de informações sobre o utilizador: nome, idade, contactos, tipos de trabalhos que este se encontra mais à vontade em realizar, número de trabalhos realizados, as classificações/comentários, e os últimos três trabalhos e as suas respetivas classificações.

Após completar a tarefa:

Quando termina a tarefa, o cliente pode classificar ou deixar comentários em relação ao trabalho e o trabalhador poderá fazer a mesma coisa respetivamente ao cliente. O rating será uma classificação de 0 a 5. Pode também deixar uma gratificação monetária pelo trabalho prestado.

Base de dados

Diagrama



Dicionário de Dados

work_state			
Descrição	estado em que o trabalho se encontra		
Nome	descrição	Tipo valor	Tamanho
ws_work_id	Id do trabalho	INT	
ws_state_id	Id do estado	INT	

_state			
Descrição	Estado do trabalho		
Nome	descrição	Tipo valor	Tamanho
state_name	Nome do estado	VARCHAR	45

rating			
Descrição	rating que um utilizador dará a outro		
Nome	descrição	Tipo valor	Tamanho
rating_comment	commentario que o utilizador poderá deixar	VARCHAR	255
rating_rat	rating do utilizador entre 1 e 5	SMALLINT	entre 1 e 5
rating_usr1_id	utilizador a dar o rating	INT	
rating_usr2_id	utilizador a receber rating	INT	

Usr			
Descrição	Utilizador		
Nome	descrição	Tipo valor	Tamanho
usr_name	Nome	VARCHAR	255
usr_email	Email do user	VARCHAR	255
usr_njobs	Numero de trabalhos efectuados	INT	
usr_avg_rating	Rating média do utilizador	smallINT	entre 1 e 5
usr_loc	Localização geografica	POINT	
usr_password	Password	VARCHAR	256

usrwork			
Descrição	Ligação entre o utilizador e o trabalho		
Nome	descrição	Tipo valor	Tamanho
uw_usr_id	Chave estrangeira dos utilizadores	INT	
uw_work_id	Chave estrangeira do trabalho	INT	
uw_usrcreate	Boolean se o usr foi o criador	boolean	

work			
Descrição	Trabalho		
Nome	descrição	Tipo valor	Tamanho
work_loc	Localização geografica do trabalho	POINT	
work_pricehr	preço hora escolhido pelo utilizador	numeric(5,2)	
work_tip	Gorjeta escolhida pelo utilizador	numeric(5,2)	
work_starting	data do inicio de trabalho	date	
work_finished	data em que o trabalho acabou	date	
work_wt_id	Id do tipo de trabalho	INT	
work_price	custo final do trabalho	numeric(5,2)	

worktype			
Descrição	Tipo de trabalho		
Nome	descrição	Tipo valor	Tamanho
wt_name	Nome do trabalho	VARCHAR	255
wt_avgprice_hr	preço hora recomentado	numeric(5,2)	

report			
Descrição	Reports que um utilizador faz		
Nome	descrição	Tipo valor	Tamanho
reported_usr_id	Chave estrangeira do utilizador reportado	INT	
report_comment	Razão do report	VARCHAR	255
report_usr_id	Chave estrangeira do utilizador que reportou	INT	
report_rc_id	Chave estrangeira para a categoria do report	INT	

report_category			
Descrição	Categorias para o report		
Nome	descrição	Tipo valor	Tamanho
rc_name	nome da categoria	VARCHAR	255

API

REST

UsersController (/api/users)

Get all users
Get all users information except the password **(THIS IS ONLY USED FOR DEBUG AND WILL NEVER BE USED IN THE APP)**

/api/users (**get**)

success:

```
[
    {
        "id": 14,
        "name": "Luis Jose Maria",
        "email": "ljm@sapo.pt",
        "jobnumber": 84,
        "rating": 3
    },
    …
]
```

Error:
No message available

Example:

```
public Iterable<User> getUsers() {

    Logger.info("Sending all units");
    return userRepository.findAll();
}
```

Add user
Adds a user when a user signs in

/api/users **(POST)**

Post example:

```
{
        "name": "name",
        "email": "email",
        "jobnumber":0,
        "password":"password"
}
```

When adding a new user on the app, (for debug purposes it's irrelevant), "jobnumber" must always be 0

Success:

```
{
    "id": id,
    "name": "name",
    "email": "email",
    "jobnumber":0,
    "rating": null
}
```

Error:
No message available

Example:

```
@PostMapping(path = "", produces = MediaType.APPLICATION_JSON_VALUE)
    public User saveUsr(@RequestBody User usr) {
        Logger.info("User named: "+usr.getName()+" saved");
        return userRepository.save(usr);
    }
```

Remove a user
Removes a specific user

/api/users/:id **(DELETE)**

Parameters:
Id: Id of the user

Success:

Error:
"No value presente"

Example:

```
@DeleteMapping(path="/{usrid}", produces = MediaType.APPLICATION_JSON_VALUE)
    public void delete(@PathVariable("usrid") int usrid){
        User user;
        user = userRepository.findById(usrid).get();
        String name = user.getName();

        Logger.info("Delete user "+name);
        userRepository.deleteById(usrid);
    }
```

Get a user
Removes a specific user

/api/users/:id **(get)**

Parameters:
Id: Id of the user

Success:

```
{
    "id": 18,
    "name": "Apolonia Casimiro",
    "email": "AC@gmail.com",
    "jobnumber": 6,
    "rating": 5
}
```

Error:
null

Example:

```
@GetMapping(path = "/{usrid}", produces = MediaType.APPLICATION_JSON_VALUE)
    public Optional<User> getUser(@PathVariable("usrid") int usrid){
        Logger.info("User with id: "+ usrid + " given");
        return userRepository.findById(usrid);
    }
```

Get a work's creator
Receives the id of a job and send back the owner

/api/users/owner/:id **(get)**

Parameters:
Id: Id of a job

Success:

```
{
    "id": 14,
    "name": "Luis Jose Maria",
    "email": "ljm@sapo.pt",
    "jobnumber": 84,
    "rating": 3
}
```

Error:
No message available

Example:

```
@GetMapping(path = "/owner/{id}", produces = MediaType.APPLICATION_JSON_VALUE)
    public Optional<User> getowner(@PathVariable("id") int id) throws
NotFoundException{
        Logger.info(id+ "");
        Optional<getownerview> _ownerid = userRepository.getowneridi(id);
        if(_ownerid.isEmpty()){
            throw new NotFoundException();
        }else{
            getownerview _owner = _ownerid.get();
             return userRepository.findById(_owner.getownerid());
        }

    }
```

WorkController(api/Work)

Get all AVAILABLE jobs for display on the map
This will send all jobs that have no starting date, and will only send the id, location and type

/api/work **(get)**

Success:

```
[
    {
        "type": "Limpeza",
        "lat": 38.7115487417554,
        "lon": -9.159549968757467,
        "id": 2
    },…
]
```

Error:
No message available

Example:

```
@GetMapping(path = "", produces = MediaType.APPLICATION_JSON_VALUE)
    public Iterable<Workmapview> getworks(){

        Logger.info("sending all jobs");

        return workrepository.workmapshow();
    }
```

Get all information of a specific work
Send all the information of a specific work

/api/work /:id **(get)**

Parameters:
Id: Id of a work

Success:

```
{
    "id": 2,
    "pricehr": 23.0,
    "tip": null,
    "started_time": null,
    "finished_time": null,
    "typeid": 2,
    "lat": 38.7115487417554,
    "lon": -9.159549968757467
}
```

Error:
No message available

Example:

```
@GetMapping(path = "/{id}", produces = MediaType.APPLICATION_JSON_VALUE)
    public Optional<Work> getwork(@PathVariable("id") int id) {
        Logger.info("Sending "+ id);
        return workrepository.findById(id);
    }
```

Get the type of a work
Gives the type of a specified work

/api/work/type/:id **(get)**

Parameters:
Id: Id of a work

Success:

```
{
    "name": "Limpeza"
}
```

Error:
No message available

Example:

```
@GetMapping(path = "/type/{id:[2-5]}", produces = MediaType.APPLICATION_JSON_VALUE)
    public gettypeView gettype(@PathVariable("id") int id){


        return workrepository.gettype(id);
    }
```

Enquadramento de outras Unidades Curriculares

Bases de Dados - Para armazenar dados, relativos às informações de cada utilizador, tipos de trabalho, entre outros.

Programação móvel - Criação da aplicação para android

Programação orientada a objetos - Utilização da linguagem Java para fazer ligação entre a base de dados e a aplicação, e processamento de dados.

Competências Comunicacionais - Criação e preparação da Apresentação da nossa Aplicação

Tecnologias a utilizar

Utilização de computadores com os seguintes programas:

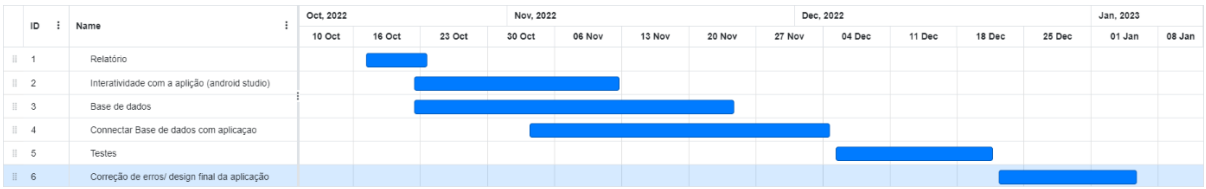
- Android Studio,
- VSCode,
- Postgres com pgAdmin 4
- GitHub
- ClickUp
- StarUML

Telemóvel com android

IPad com as aplicações:

- GoodNotes
- Procreate

Planeamento e calendarização



Francisco

week 1- base de dados

week 2- android studio, conectar base de dados com app

week 3- android studio, base de dados

week 4- conectar base de dados, base de dados

week 5- conectar base de dados

week 6- testes

week 7- testes

week 8-

week 9- Correção de erros

week 10- finalizar o design da app

week 11- finalizar o design da app

Gonçalo

week 1- android studio

week 2- base de dados, conectar base de dados com app

week 3- android, conectar base de dados com app

week 4- conectar base de dados, base de dados

Gustavo

week 1- android studio

week 2- base de dados, conectar base de dados com app

week 3- android, base de dados

week 4- conectar base de dados, base de dados

week 5- conectar base de dados

week 6-testes

week 7-testes

week 8-testes

week 9- Correção de erros

week 10- finalizar o design da app

week 11- finalizar o design da app

week 5- conectar base de dados

week 6- testes

week 7- testes

week 8- testes

week 9- Correção de erros

week 10- finalizar o design da app

week 11- finalizar o design da app

Bibliografia:

https://logicaldollar.com/odd-jobs-app/

https://www.google.pt/maps

https://www.fiverr.com/

www.taskrabbit.pt

https://staruml.io/

https://github.com/

https://code.visualstudio.com/

clickup.com/

https://www.postgresql.org/

https://www.olx.pt/