2017
LEE

Sistema de Votação Online

PROJECTO IRC

AFONSO MURALHA, JOÃO GALAMBA, NUNO MIGUEL MACARA

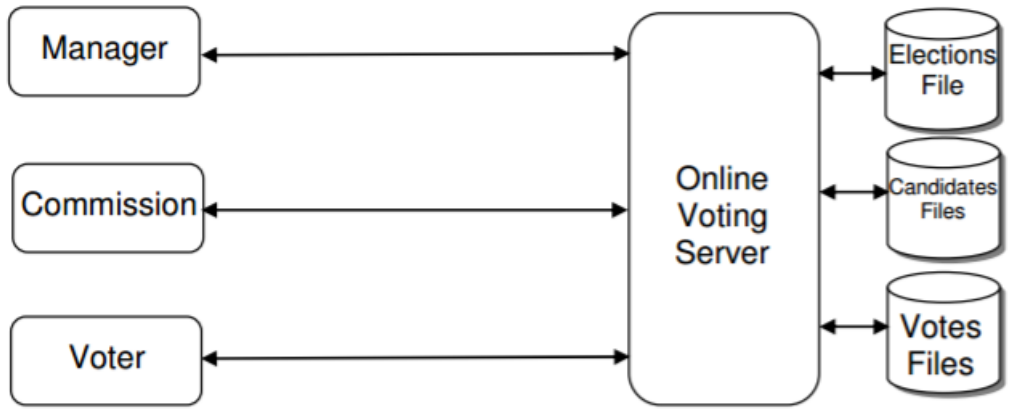
Conteúdo

Objetivos

Implementar um sistema de votação online

Especificação

O sistema de votação é baseado numa arquitetura cliente-servidor, onde existe um sistema *"One-client login system"* implementado de raiz para destribuir a todos os clientes.



Existem 3 tipos de clientes:

- Um cliente de gestão de votações, **manager**
- Um cliente para a aprovação de candidatos, **comissão** eleitoral
- Um cliente para servir aos leitores, **eleitor**

O cliente de gestão pode criar uma nova votação indicando o nome da mesma, listar todas as votações bem como abrir o período de votação. Depois do período de votação acabar o gestor pode fechar a votação e lançar os resultados. Uma votação fechada não poderá ser aberta novamente.

A comissão eleitoral pode adicionar candidatos a uma votação existente que tenha sido criada pelo gestor se esta não tenha sido aberta. A comissão pode também listar os candidatos existentes para cada votação.

Os eleitores podem votar nas votações abertas e consultar os resultados das votações. Se uma votação estiver aberta, o eleitor poderá votar apenas uma vez num único candidato. Para votar, o eleitor tem de fornecer o número do seu cartão de cidadão, o nome da votação e no candidato em que pretende votar. Se uma votação fechar, os eleitores deixarão de poder votar, mas poderão consultar os resultados.

O servidor deverá gerir as votações existentes, os candidatos de cada votação bem como o estado de cada votação. O servidor deverá armazenar toda a informação relacionada com as votações em ficheiros de texto. O servidor deverá também enviar confirmações aos clientes por cada mensagem que envia.

O formato dos ficheiros de texto implementados terá a seguinte estrutura:

"votacoes.txt"	"LEE.candidatos"	"LEE.votos"	"LEE.cc"
LEE 1 LETI 0 LEGI 2	Aluno1 Aluno2 Aluno3	Aluno1 25 Aluno 2 1 Aluno 3 10	12345678 88888888 25357845

O ficheiro votacoes.txt guarda em cada linha uma votação seguida do estado (criado=0, aberto=1, fechado=2), separado por um espaço. Para cada votação existe um ficheiro com o nome da votação, com a extensão ".candidatos", listando os candidatos da respetiva votação, um por cada linha, um ficheiro com o nome da votação, com a extensão ".votos" listando os candidatos e seus respetivos votos como exemplificado acima e finalmente, existe o ficheiro com a extensão ".cc" que indica todos os utilizadores que já votaram, isto para não existir mais que um voto por cada eleitor na respectiva votação.

O servidor está preparado para continuar as eleições gravadas nos ficheiros. Se por alguma razão houver alguma incoerência nos ficheiros o servidor é capaz de detctar isso e informar o manager e o comission que os ficheiros estão corrompidos. O manager poderá usar o comando "*cleandir*" para apagar todos os ficheiros e criar novos.

A distribuição de trabalho foi feita igualmente por todos os membros do grupo, representado pela tabela abaixo:

Afonso Muralha	Nuno Miguel Macara	João Galamba
Manager	Comissão	Eleitores
Servidor		

O trabalho foi demonstrado da semana de 17-21 de abril e notou-se que apesar de estar tudo implementado, existiam alguns bugs no que diz respeito à especificação dos erros. Na demonstração final o programa deverá estar livre de qualquer tipo de erros e/ou bugs.

Detalhes e Especificação

Implementação

O projecto será desenvolvido na linguagem de programação Python (versão 3) sendo todos os ficheiros representados com a extensão .py e poderão ser executados com o comando "*python ficheiro.py*" num terminal Linux/Unix.

O sistema funciona executando uma instância do programa servidor. Os clientes comunicam com o servidor abrindo o ficheiro *client.py* e fazendo *login* especificando a sua função (voter, comission ou manager). O número de utilizadores ligados como manager ou comission é apenas de um por servidor, ao contrário dos utilizadores eleitores que o numero de eleitores ligados depende única e exclisivamente da capacidade da máquina que está a correr o servidor.

Especificação do protocolo de comunicação

Protocolo de transporte

Cada cliente comunica com o servidor por uma socket em que o porto de comunicação é selecionado ao executar o programa. Foi escolhido o protocolo UDP visto que para um projeto desta dimensão não se justificava a comunicação por TCP, apesar deste ser fiável. Como o servidor responde a todas as mensagens enviadas pelos clientes, dá-nos de certa forma alguma segurança apesar de este sistema não ser infalível.

Sistema de *login*

Utilização

Como referido anteriormente, o sistema tem implementado um gestor de clientes que premite ser usado apenas um programa para todos os clientes, sejam estes voter, manager ou comission. Para se registar, o cliente apenas tem de introduzir a sua função, e se for aceite pelo servidor, poderá utilizar os comandos indicados à sua função. Enquanto o utilizador não intruduzir um utilizador válido, este não poderá executar nenhum comando.

Protocolo de autenticação

Para se efectuar o registo, o cliente envia um pedido de autenticação ao servidor, enviado a mensagem “LOGFUNC “ seguida da especificação do cliente. Se for válida, o servidor responde com a mensagem “LOGACCEPT”, se não responde com “ERROR_USERTAKEN” e o cliente envia um novo pedido de auntenticação. Este ciclo repete-se até o servidor aceitar o pedido de autenticação. Nenhuma destas mensagens são visiveis para o utilizador, apesar de poderem ser facilmente capturadas por uma sonda de captura de pacotes, visto que nenhuma das mensagens são encriptadas. Isto pode resultar numa grande falha de segurança, mas visto que não é o objectivo do projecto, não foi implementado um sistema de encriptação. Consultar também o diagrama abaixo.

Formato das mensagens

A estrutura das mensagens segue a regra em que o comando é seguido dos argumentos utilizados pelo utilizador (ex: info votacao1), para aceder a informações dos comandos é possível utilizar o único argumento -help que mostra a descrição e opções de cada comando.

Comandos e respostas

Eleitor:

```
Envia:
        info <nome_votacao>
    Recebe:
        "<nome_votacao> aberta/fechado/nao_existente"


    Envia:
        voto <nome_votacao> <info_pessoal> <nome_candidato>
        (justificacao: "Eu sou ... quero votar para ... quero votar em ...)
    Recebe:
        Se <nome_votacao> <info_pessoal> <nome_candidato>, validos (votacao
aberta e existente):
            Recebe: "Voto contabilizado"
        Se um ou mais argumentos estejam invalidos (votacao fechada e/ou nao
existente):
            Recebe: Erro/os especifico ao/s argumento/os

    Envia:
        resultado <nome_votacao>

    Recebe:
        Se <nome_votacao> for valido (fechado e existente):
            Recebe: Resultado específico da votacao
        Se <nome_votacao> for invalido (aberto ou nao existente):
            Recebe: Erro específico ao estado da votacao

    Envia:
        logout

    Recebe:
         “Successfully logged out…”


    Envia:
        outro
    Recebe: "Comando Invalido"
```

Comissão:

```
Envia:
        info <nome_votacao>
    Recebe:
        "<nome_votacao> aberta/fechado/nao_existente"
    Envia:
        info
    Recebe: Info sobre todas as votacões e respectivos estados



    Envia:
        candidatos <nome_votacao>
    Recebe: Informacao sobre os candidatos da respectiva votacao

    Envia:
        candidatos
    Recebe: Informacao sobre todos os candidatos


    Envia:
        adicionar <nome_candidato> <nome_votacao>
    Recebe:
        Se <nome_candidato> <nome_votacao> validos (votacao valida e criada)
            Recebe: "Candidato <nome_candidato> adicionado a votacao
<nome_votacao>"
        Se <nome_candidato> <nome_votacao> invalido (votacao
invalida/aberta/fechada)
            Recebe: Erro respectivo ao argumento

Envia:
        logout
    Recebe:
        “Successfully logged out…”


    Envia:
        outro
    Recebe: "Comando Invalido"
```

Manager:

```
Envia:
        cria_eleicao <nome_votacao>
    Recebe:
        Se <nome_votacao> ja existir
            Recebe: "Eleicao ja criada"
        Se <nome_votacao> valido
            Recebe: "Eleicao criada"


    Envia:
        info <nome_votacao>
    Recebe:
        "<nome_votacao> aberta/fechado/nao_existente"
    Envia:
        info
    Recebe: Info sobre todas as votacões e respectivos estados


    Envia:
        abre <nome_votacao>
    Recebe:
        Se <nome_votacao> valido (criada)
            Recebe: "<nome_votacao> iniciada"
        Se <nome_votacao> invalido (fechada/aberta/nao existente)
            Recebe: Erro respectivo


    Envia:
        fecha <nome_votacao>
    Recebe:
        Se <nome_votacao> valido (aberta)
            Recebe: "<nome_votacao> fechada"
        Se <nome_votacao> invalido (fechada/nao existente)
            Recebe: Erro respectivo

    Envia:
        cleandir
    Recebe:
        “Directory cleaned…”
    Envia:
        logout
    Recebe:
        “Successfully logged out…”
    Envia:
        killserver
    Recebe:
        “Server killed…



    Envia:
        outro
    Recebe: "Comando Invalido"
```

Diagramas Temporais

Diagrama de Comunicação Normal

1. cria_votacao eleicao1
2. “Votacao criada”
3. adicionar candidatoA eleicao1
4. “Candidato candidatoA adicionado a votacao eleicao1”
5. adicionar candidatoB eleicao1
6. “Candidato candidatoB adicionado a votacao eleicao1”
7. abre eleicao1
8. “eleicao1 iniciada”
9. info eleicao1
10. “eleicao1 aberta”
11. voto eleicao1 <cc1> candidatoA
12. “voto contabilizado”
13. voto eleicao1 <cc1> candidatoB
14. “voto invalido”
15. fecha eleicao1
16. “eleicao1 fechada”
17. resultado eleicao1
18. <lista de resultados>

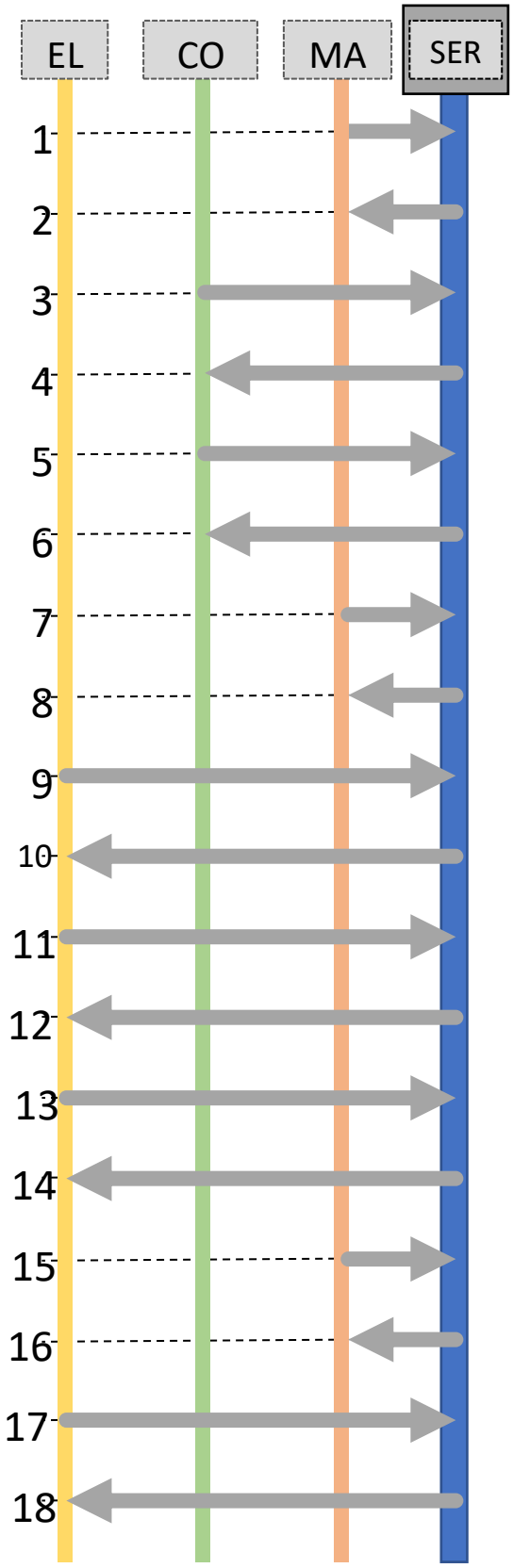
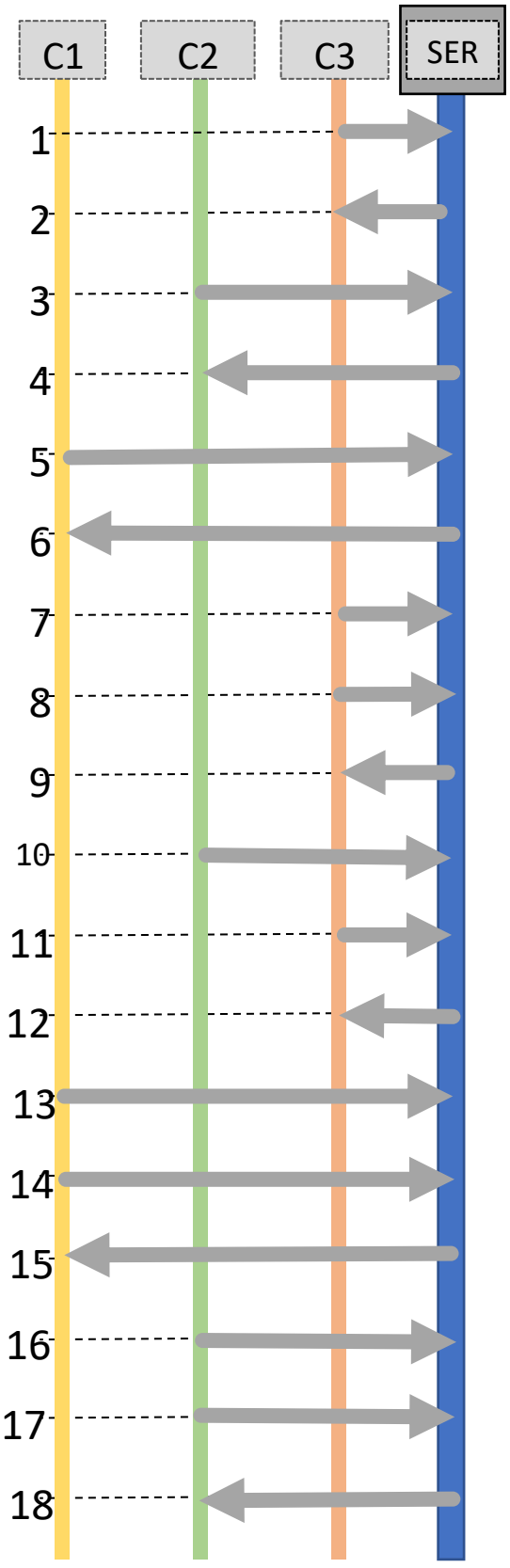


Diagrama de Login

1. LOGFUNC voter
2. LOGACCEPT (C3 é voter)
3. LOGFUNC manager
4. LOGACCEPT (C2 é manager)
5. LOGFUNC comission
6. LOGACCEPT (C1 é comission)
7. Logout
8. LOGFUNC manager
9. ERROR_USERTAKEN
10. Logout
11. LOGFUNC manager
12. LOGACCEPT (C3 é manager)
13. Logout
14. LOGFUNC voter (C1 é voter)
15. LOGACCEPT
16. Logout
17. LOGFUNC voter
18. LOGACCEPT (C2 é voter)



Nota: as mensagens demonstradas nesta página demonstram **apenas a comunicação cliente-servidor.**

Respostas de Erro

Nesta implementação, o servidor que gere todo o sistema está programado para que possa receber comandos incorrectos, aos quais responde com uma mensagem de erro que especifica a anomalia.

Todas as mensagens de erro são específicas a cada caso, para que o utilizador possa corrigir o seu comando.

Erros esses são:

Eleitor:

O eleitor vota numa votação inexistente ou fechada.

O eleitor vota num candidato inexistente.

O eleitor tenta votar mais que uma vez.

O eleitor pede o resultado de uma votação aberta ou inexistente.

O eleitor envia um comando desconhecido ao servidor.

Comissão:

A comissão pede informação sobre um candidato inexistente.

A comissão adiciona um candidato a uma votação inválida/aberta/fechada.

A comissão envia um comando desconhecido ao servidor.

Ficheiros corrompidos

Manager:

O manager abre uma votação inválida.

O manager fecha uma votação fechada/inexistente.

O manager envia um comando desconhecido ao servidor.

Ficheiros corrompidos

Informações Adicionais

O projeto tem o suporte da plataforma GitHub para facilitar a gestão de versões. O repositório encontra-se privado, sendo que apenas os membro do grupo têm permissão para ver e editar. Se desejado o docente também pode obter permissões para visualizar a página, tendo apenas de solicitar a um dos membros do grupo