UNIVERSIDADE ABERTA
www.uab.pt

UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO

Mestrado em Engenharia Informática e Tecnologia Web

Integração de Sistemas
(22295)
Ano letivo 2025/2026

TRABALHO PRÁTICO DE INTEGRAÇÃO DE SISTEMAS

DESENVOLVIMENTO DE APLICAÇÃO DE VOTAÇÃO ONLINE

André Sousa
1300012@estudante.uab.pt

GIT https://github.com/AndreMacielSousa/VotingSystem-App

Índice

Índice de Ilustrações

DESENVOLVIMENTO DE APLICAÇÃO DE VOTAÇÃO ONLINE

1. Introdução

A integração de sistemas constitui um dos desafios centrais no desenvolvimento de soluções informáticas contemporâneas, em particular em contextos institucionais que exigem interoperabilidade, fiabilidade, segurança e alinhamento com requisitos legais e organizacionais. A crescente adoção de arquiteturas distribuídas e orientadas a serviços tem vindo a reforçar a importância de mecanismos claros de separação de responsabilidades, contratos bem definidos e governação técnica adequada, sobretudo em sistemas críticos para a sociedade.

Neste contexto, os sistemas de votação eletrónica representam um domínio particularmente exigente do ponto de vista da Integração de Sistemas. Para além da necessidade de integração entre múltiplos serviços e entidades independentes, estes sistemas devem conciliar requisitos potencialmente antagónicos, nomeadamente a validação rigorosa dos votantes, a garantia de que cada eleitor vota uma única vez e a preservação absoluta do anonimato do voto. A falha em qualquer um destes aspetos compromete a credibilidade, a legitimidade e a confiança no processo eleitoral.

O presente trabalho insere-se no âmbito da unidade curricular **Integração de Sistemas**, do Mestrado em Engenharia Informática e Tecnologia Web, e tem como objetivo a análise, conceção e implementação de um protótipo académico de aplicação de votação eletrónica, baseado em serviços distribuídos com interface gRPC. O trabalho parte de um conjunto de serviços eletrónicos disponibilizados em contexto de *mock*, que simulam as funções essenciais de um sistema de votação, permitindo explorar de forma prática os desafios associados à integração, aos contratos de serviço e à robustez das soluções desenvolvidas.

De acordo com o enunciado do trabalho prático, o desenvolvimento do projeto encontra-se organizado em três tarefas complementares. A **Tarefa 1** incide sobre a proposta de uma arquitetura simples para um sistema de votação eletrónica que assegure simultaneamente o anonimato do voto e a validação dos votantes, explorando princípios clássicos de separação de responsabilidades e confiança distribuída, bem como o possível papel dos sistemas da Administração Pública nesse contexto. A **Tarefa 2** foca-se na experimentação direta dos serviços gRPC disponibilizados, recorrendo à ferramenta *grpcurl*, com o objetivo de identificar limitações técnicas relevantes quando se pretende construir uma aplicação robusta e fiável. Por fim, a **Tarefa 3** consiste na conceção e implementação de uma aplicação de votação eletrónica que materializa o processo definido no enunciado, justificando as opções tecnológicas adotadas, descrevendo os casos de uso e avaliando criticamente as limitações da solução desenvolvida.

O trabalho adota uma abordagem prática e reflexiva, combinando implementação técnica com análise crítica das decisões tomadas e das limitações observadas. A utilização de ferramentas de apoio baseadas em inteligência artificial foi encarada como um instrumento auxiliar ao processo de desenvolvimento, não substituindo o raciocínio crítico nem a validação empírica das soluções adotadas. O foco principal reside, assim, na compreensão dos problemas de integração de sistemas, na análise dos contratos de serviço e na avaliação da adequação das soluções propostas face aos requisitos funcionais e não funcionais do domínio em estudo.

A estrutura do relatório reflete esta organização. Após a presente introdução, o documento apresenta a proposta arquitetural do sistema de votação eletrónica, seguida da análise crítica dos serviços gRPC disponibilizados. De seguida, descreve-se a aplicação desenvolvida, os respetivos casos de uso, testes

realizados e decisões de implementação. O relatório termina com uma discussão integrada dos resultados obtidos e uma conclusão final, onde são sintetizadas as principais aprendizagens e identificadas possíveis evoluções futuras.

2. Arquitetura de um Sistema de Votação Eletrónica com Anonimato

TAREFA 1

2.1. Enquadramento do problema

O desenho de sistemas de votação eletrónica constitui um desafio particularmente exigente no domínio da Integração de Sistemas, na medida em que obriga à conciliação de dois requisitos fundamentais e potencialmente antagónicos: **a validação rigorosa dos votantes** e **a garantia do anonimato do voto**. Um sistema credível deve assegurar que apenas eleitores elegíveis participam no processo e que cada eleitor vota uma única vez, sem que seja possível, em qualquer momento, estabelecer uma relação entre a identidade civil do cidadão e a opção de voto expressa.

Partindo da prompt *"como criaria um sistema de votação eletrónica que garantisse o anonimato do voto mas validasse os votantes"*, foi proposta uma arquitetura simples que assenta em princípios clássicos de separação de responsabilidades, desacoplamento funcional e distribuição de confiança, amplamente discutidos na literatura sobre arquiteturas orientadas a serviços e integração de sistemas.

2.2. Proposta de arquitetura simples

A arquitetura proposta baseia-se num princípio estruturante: **nenhuma entidade do sistema deve, isoladamente, deter informação suficiente para associar identidade e voto**. Para tal, o sistema é organizado em torno de três entidades funcionais distintas, cada uma com responsabilidades bem delimitadas.

2.2.1 Entidades funcionais

- **Autoridade de Registo (AR)**

  A Autoridade de Registo é responsável pela validação da identidade e elegibilidade do eleitor. Nesta fase, o eleitor identifica-se através de mecanismos oficiais e, após validação, é-lhe emitida uma **credencial de voto** única. Esta credencial não contém informação identificativa direta e não permite, por si só, inferir a identidade do eleitor. A AR não participa no processo de votação nem tem acesso à escolha efetuada.

- **Autoridade de Votação (AV)**

  A Autoridade de Votação recebe o voto eletrónico juntamente com a credencial emitida pela AR. A sua função consiste em verificar a validade da credencial e garantir que esta não foi previamente utilizada, assegurando assim que cada eleitor vota apenas uma vez. A AV não dispõe de dados de identificação civil dos eleitores, operando exclusivamente sobre credenciais pseudónimas.

- **Autoridade de Apuramento (AA)**

A Autoridade de Apuramento é responsável pela contabilização dos votos e pela divulgação dos resultados eleitorais. O apuramento é realizado sobre votos recolhidos de forma anónima, sem acesso a credenciais ou dados que permitam reidentificação[1] dos eleitores.

Esta divisão funcional traduz uma abordagem de confiança distribuída, reduzindo significativamente o risco de comprometer simultaneamente a privacidade do eleitor e a integridade do processo.

2.3. Fluxo do processo de votação

O funcionamento do sistema pode ser descrito em três fases sequenciais:

1. **Fase de registo**

   O eleitor identifica-se junto da Autoridade de Registo, que valida a sua elegibilidade com base em dados oficiais. Após validação, é emitida uma credencial de voto única e não reutilizável.

2. **Fase de votação**

   O eleitor utiliza a credencial para aceder à Autoridade de Votação, consultar a lista de candidatos e submeter o voto. A AV valida a credencial e, caso seja válida e não utilizada, aceita o voto e regista a credencial como usada.

3. **Fase de apuramento**

   A Autoridade de Apuramento recolhe os votos registados e procede à sua contabilização, garantindo a transparência e integridade dos resultados sem comprometer o anonimato.

2.4. Integração dos sistemas da Administração Pública

Os sistemas da Administração Pública (AP) podem desempenhar um papel central nesta arquitetura, desde que respeitado o princípio da separação de funções e da minimização de dados.

2.4.1 Administração Pública como suporte à Autoridade de Registo

A Autoridade de Registo pode integrar serviços já existentes na AP, nomeadamente sistemas de **identificação eletrónica** (como o Cartão de Cidadão) e **bases de dados de recenseamento eleitoral**. Estes sistemas permitem validar de forma fiável a identidade e elegibilidade dos eleitores, sem necessidade de expor essa informação às restantes entidades do sistema.

2.4.2 Administração Pública e governação da votação

Na Autoridade de Votação, a intervenção da AP deve centrar-se na definição de políticas de governação, auditoria e monitorização do serviço, assegurando que os mecanismos técnicos cumprem requisitos legais e institucionais, sem acesso a dados de identidade.

2.4.3 Transparência e apuramento

No apuramento, a Administração Pública pode garantir transparência e legitimidade institucional, assegurando que os resultados são verificáveis e auditáveis. A separação entre votação e apuramento reforça a confiança no sistema e reduz o risco de manipulação ou abuso de informação.

[1] **Reidentificação** ocorre quando, direta ou indiretamente, é possível estabelecer novamente a ligação entre um registo técnico (por exemplo, um voto, um identificador, um token ou um conjunto de metadados) e a identidade real do seu titular.

2.5. Consideração crítica

A arquitetura apresentada é intencionalmente simples, adequada ao contexto académico e ao objetivo de ilustrar princípios fundamentais de integração de sistemas. Embora sistemas de votação eletrónica reais exijam mecanismos criptográficos e organizacionais mais avançados, a proposta demonstra que a combinação entre **separação de responsabilidades**, **credenciais de uso único** e **integração controlada de sistemas da Administração Pública** constitui uma base sólida para garantir simultaneamente anonimato e validação dos votantes.

3. Análise dos Serviços gRPC e Limitações à Robustez da Integração

TAREFA 2

3.1. Contexto e objetivo

De acordo com o enunciado, esta tarefa consiste em **testar os serviços gRPC disponibilizados (mockups)** através da ferramenta *grpcurl* e, com base nos resultados e na documentação do projeto, **identificar limitações relevantes** quando o objetivo é construir uma aplicação de votação eletrónica robusta.

O endpoint indicado para testes é **ken01.utad.pt:9091** e são fornecidos exemplos de invocação para emissão de credencial, consulta de candidatos e submissão de voto (credencial válida e inválida).

3.2. Testes executáveis com *grpcurl* (comandos do enunciado)

3.2.1 Emissão de credencial (AR)

O serviço de registo emite uma credencial com base no número de Cartão de Cidadão (no mock, o CC é “aceite sempre”), devolvendo credenciais válidas ou inválidas de forma probabilística.

```
PS C:\work\VotingSystem> '{"citizen_card_number":"123456789"}' | grpcurl -insecure -proto Protos/voter.proto -d "@" ken01.utad.pt:9091
voting.VoterRegistrationService/IssueVotingCredential
{
  "isEligible": true,
  "votingCredential": "CRED-ABC-123"
```

Figura 1 - Teste 1 – Emissão de credencial (IssueVotingCredential)

Foi testado o serviço VoterRegistrationService/IssueVotingCredential através de *grpcurl*, submetendo o identificador "123456789" como número de Cartão de Cidadão. O serviço respondeu com isEligible = true e devolveu a credencial CRED-ABC-123. Este resultado confirma a operacionalização do fluxo de emissão de credenciais no *mock* e valida a disponibilidade do endpoint gRPC. Contudo, o teste foi executado com -insecure, o que significa que não foi efetuada validação de confiança TLS, evidenciando que, num cenário de produção, seriam necessários mecanismos formais de segurança do canal e autenticação entre componentes. Adicionalmente, por se tratar de um serviço simulado, a elegibilidade devolvida não traduz uma verificação efetiva de identidade/recenseamento, mas sim um comportamento de teste orientado ao consumo do contrato gRPC.

3.2.2 Consulta da lista de candidatos (AV)

```
PS C:\work\VotingSystem> grpcurl -insecure -proto Protos/voting.proto ken01.utad.pt:9091 voting.VotingService/GetCandidates
{
  "candidates": [
    {
      "id": 1,
      "name": "Candidato A"
    },
    {
      "id": 2,
      "name": "Candidato B"
    },
    {
      "id": 3,
      "name": "Candidato C"
    }
  ]
}
PS C:\work\VotingSystem>
```

Figura 2 - Teste 2 – Consulta de candidatos (GetCandidates)

Foi testado o serviço VotingService/GetCandidates através de *grpcurl*, recorrendo ao contrato definido em voting.proto. O serviço respondeu com uma lista de três candidatos, identificados pelos IDs 1, 2 e 3, respetivamente "Candidato A", "Candidato B" e "Candidato C". Este resultado confirma a correta disponibilização do catálogo de candidatos e a capacidade do cliente gRPC em consumir o serviço de consulta. A existência de identificadores numéricos bem definidos é essencial para garantir a coerência entre a fase de consulta e a fase de votação, reduzindo erros de submissão por referência a candidatos inexistentes.

Limitações identificadas a partir deste teste

Estas limitações **derivam diretamente do que foi observado**, sem extrapolações indevidas:

1. **Ausência de metainformação temporal ou de versão**

   O serviço não indica qualquer versão do catálogo nem validade temporal, o que pode originar inconsistências se a lista de candidatos for alterada entre a consulta e a votação.

2. **Catálogo estático e simplificado (mock)**

   Os candidatos devolvidos são fixos e genéricos, o que é adequado para testes mas insuficiente para um cenário real, onde seriam necessários dados adicionais (por exemplo, identificadores oficiais, validações de integridade ou assinaturas).

3. **Segurança do canal não garantida**

   Tal como no Teste 1, o uso de -insecure indica que o canal não é validado criptograficamente, sendo aceitável em laboratório, mas inadequado para produção.

Estas observações reforçam que, numa aplicação robusta, a consulta de candidatos deve ser acompanhada por mecanismos de versionamento, validação e governação do serviço.

3.2.3 Submissão de voto com credencial válida (AV)

```
PS C:\work\VotingSystem> '{"voting_credential":"CRED-ABC-123","candidate_id":1}' | grpcurl -insecure -proto Protos/voting.proto -d "@"
 ken01.utad.pt:9091 voting.VotingService/Vote
{
  "success": true,
  "message": "Voto registado para o Candidato A."
}
PS C:\work\VotingSystem>
```

Figura 3 - Teste 3.1 – Submissão de voto com credencial válida

Foi testado o serviço VotingService/Vote recorrendo à credencial válida CRED-ABC-123, previamente obtida no Teste 1, e ao identificador do candidato candidate_id = 1. O serviço respondeu com success = true, confirmando o registo do voto para o "Candidato A". Este resultado demonstra que o serviço aceita corretamente votos submetidos com credenciais válidas e que a coerência entre a fase de consulta de candidatos e a fase de votação é mantida. No entanto, a resposta evidencia também a simplicidade do *mock*, uma vez que não são fornecidas informações adicionais sobre persistência, auditoria ou garantias de integridade transacional do voto.

Limitações observadas neste sub-teste

Estas limitações decorrem **diretamente do comportamento observado**:

1. **Feedback funcional minimalista**

   A resposta limita-se a indicar sucesso e uma mensagem textual, sem fornecer identificadores de transação ou comprovativos auditáveis do registo do voto.

2. **Ausência de garantias explícitas de idempotência**

   Não é possível inferir, a partir da resposta, como o serviço lida com reenvios do mesmo pedido (por exemplo, em caso de falha de rede após submissão).

3. **Segurança do canal não verificada**

   Tal como nos testes anteriores, o uso de -insecure indica que não há validação criptográfica do canal, o que é aceitável em ambiente de teste mas insuficiente para produção.

```
PS C:\work\VotingSystem> '{"voting_credential":"CRED-ABC-123","candidate_id":1}' | grpcurl -insecure -proto Protos/voting.proto -d "@"
 ken01.utad.pt:9091 voting.VotingService/Vote
{
  "success": true,
  "message": "Voto registado para o Candidato A."
}
PS C:\work\VotingSystem> '{"voting_credential":"CRED-ABC-123","candidate_id":1}' | grpcurl -insecure -proto Protos/voting.proto -d "@"
 ken01.utad.pt:9091 voting.VotingService/Vote
{
  "success": true,
  "message": "Voto registado para o Candidato A."
}
PS C:\work\VotingSystem>
```

Figura 4 - Teste 3.2 – Reutilização da mesma credencial de voto

Após o registo bem-sucedido de um voto com a credencial CRED-ABC-123, foi repetida a submissão de voto utilizando exatamente a mesma credencial. Observou-se que o serviço voltou a aceitar o pedido e a responder com sucesso, evidenciando que o *mock* não implementa mecanismos de controlo de unicidade das credenciais. Esta limitação compromete o requisito de garantir que cada eleitor vota apenas uma vez e demonstra que o serviço, na sua forma atual, não mantém estado nem valida a reutilização de credenciais. Para uma aplicação robusta, seria indispensável introduzir persistência de estado, verificação de credenciais já utilizadas e garantias de idempotência no processo de votação.

3.2.4 Submissão de voto com credencial inválida (AV)

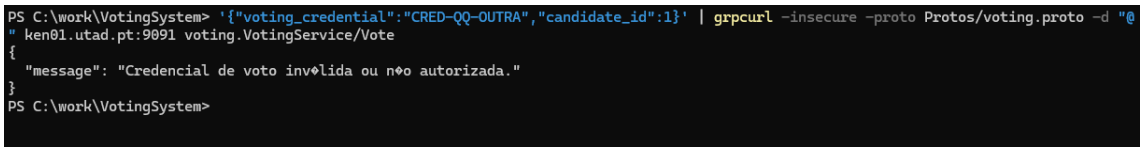

Figura 5 - Teste 3.3 – Submissão de voto com credencial inválida

Foi testada a submissão de um voto utilizando uma credencial inválida (CRED-QQ-OUTRA), não pertencente ao conjunto de credenciais aceites pelo sistema. O serviço respondeu com a mensagem "Credencial de voto inválida ou não autorizada.", confirmando que existe validação básica das credenciais apresentadas. Este comportamento demonstra que o serviço distingue credenciais reconhecidas de valores arbitrários. No entanto, a resposta não inclui um indicador explícito de falha (por exemplo, um campo booleano success = false), o que pode dificultar o tratamento uniforme de erros por parte de clientes gRPC mais robustos.

Limitações observadas a partir deste sub-teste

A partir deste resultado concreto, podem ser registadas as seguintes limitações:

1. **Contrato de erro incompleto**

   A resposta devolve apenas uma mensagem textual, sem um campo estruturado que indique explicitamente o insucesso da operação, o que reduz a clareza do contrato do serviço.

2. **Ausência de códigos de erro normalizados**

   Não são fornecidos códigos de erro ou razões formais que permitam aos clientes distinguir, de forma programática, entre diferentes causas de rejeição (credencial inválida, credencial já usada, candidato inexistente, etc.).

3.3. Principais limitações dos serviços (mock) para suportar uma aplicação robusta

A execução dos testes com *grpcurl* confirmou a operacionalidade do fluxo base (emissão de credencial, consulta de candidatos e submissão de voto). Contudo, os resultados evidenciam limitações relevantes quando se pretende evoluir para uma solução robusta, com garantias de segurança, consistência e governança, como se espera num cenário real de votação eletrónica.

3.3.1 Validação de identidade e elegibilidade não é verificável no mock

No Teste 1, a emissão de credencial devolveu isEligible = true e uma credencial válida (CRED-ABC-123). Apesar disso, o comportamento observado (e o próprio contexto de serviço simulado) não permite concluir que exista validação real de identidade/recenseamento; trata-se, sobretudo, de um mecanismo funcional para suportar testes ao consumo de credenciais. Numa aplicação real, este papel teria de ser integrado com serviços institucionais de identificação e elegibilidade.

3.3.2 Comportamento de emissão de credenciais potencialmente não determinístico

Sendo um serviço de teste, a emissão de credenciais pode produzir resultados distintos em invocações diferentes (por exemplo, credenciais não aceites posteriormente pela votação). Este tipo de

comportamento é útil para testar fluxos de erro, mas, do ponto de vista do cliente, obriga a estratégias explícitas de gestão de falhas (repetição controlada, mensagens claras e prevenção de sobrecarga), sob pena de comprometer a experiência do utilizador e a estabilidade do sistema.

3.3.3 Unicidade do voto não garantida (limitação crítica)

O resultado mais relevante foi a observação de que a **mesma credencial pode ser reutilizada com sucesso**: após uma submissão válida no Teste 3.1, a repetição com a mesma credencial voltou a ser aceite (Teste 3.3). Isto demonstra que o mock não implementa controlo de "credencial já utilizada" e, consequentemente, não garante a propriedade "um eleitor, um voto". Numa solução robusta, este controlo exige persistência de estado e validação transacional da credencial.

3.3.4 Contratos de erro pouco estruturados

No Teste 3.2, uma credencial inválida foi rejeitada com a mensagem "Credencial de voto inválida ou não autorizada.", mas a resposta não incluiu um indicador estruturado de falha (por exemplo, success=false) nem códigos normalizados de erro. Esta ausência dificulta a implementação de clientes robustos, uma vez que a lógica de tratamento de erros fica dependente de parsing textual e não de semântica contratual explícita.

3.3.5 Segurança do canal não demonstrada nos testes

Em todos os testes foi utilizado o parâmetro -insecure, o que significa que não foi efetuada validação criptográfica de confiança (certificados). Embora adequado ao laboratório, este aspeto é incompatível com requisitos de produção, onde seriam necessárias medidas como TLS devidamente configurado, autenticação forte entre serviços (por exemplo, mTLS) e mecanismos anti-abuso.

3.3.6 Consistência fim-a-fim e idempotência não são garantidas

O fluxo "emitir credencial → consultar candidatos → votar" é, na prática, um processo distribuído multi-etapa. Os outputs observados não evidenciam garantias de idempotência (por exemplo, proteção contra reenvios acidentais) nem mecanismos de rastreabilidade transacional (IDs de operação, comprovativos ou estados intermediários). Em cenários reais, falhas parciais (timeouts e duplicação de pedidos) podem conduzir a inconsistências sem medidas explícitas de desenho.

3.3.7 Observabilidade, auditoria e governança não são suportadas pelo contrato

Os serviços testados devolvem respostas funcionais mínimas (sucesso/mensagem), mas não disponibilizam elementos que suportem auditoria técnica (logs auditáveis, comprovativos verificáveis, mecanismos de verificação do conjunto de votos) nem aspectos de governança (versão de catálogo de candidatos, políticas de retenção, compatibilidade de contratos). A literatura sobre SOA e integração de sistemas sublinha que benefícios de interoperabilidade e reutilização dependem de governança e contratos estáveis para evitar fragilidade operacional.

3.4 Síntese crítica da análise dos serviços

Os testes com *grpcurl* validaram o funcionamento do fluxo base e permitiram identificar limitações determinantes do mock. Em particular, destaca-se a inexistência de controlo de unicidade de credenciais (dupla votação), a fragilidade do contrato de erros (mensagens pouco estruturadas) e a ausência de garantias de segurança e consistência distribuída. Para uma aplicação robusta, estes resultados apontam para a necessidade de persistência e validação transacional, mecanismos de segurança adequados, e práticas de governança típicas de ambientes SOA/EAI.

4. Desenvolvimento de uma Aplicação de Votação Eletrónica Baseada em gRPC

TAREFA 3

A presente tarefa tem como objetivo a conceção, implementação e validação de um protótipo funcional de aplicação de votação eletrónica, em conformidade com o processo definido no ponto 2.2 do enunciado. Em particular, esta tarefa visa: (i) justificar as tecnologias e frameworks adotadas, (ii) descrever os casos de uso da aplicação e respetivo diagrama, (iii) apresentar a implementação e os testes realizados, e (iv) refletir criticamente sobre as limitações e possíveis evoluções da solução proposta.

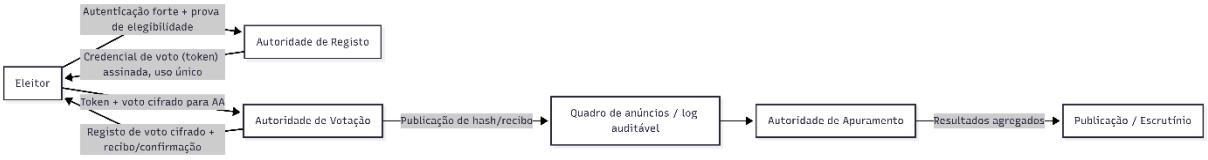

Figura 6 - Diagrama de arquitetura

4.1. Contratos gRPC disponibilizados

O desenho da aplicação desenvolvida na Tarefa 3 é diretamente orientado pelos contratos gRPC definidos nos ficheiros .proto fornecidos no âmbito do trabalho. Estes contratos estabelecem, de forma explícita e formal, as operações disponibilizadas pelos serviços remotos, os tipos de dados trocados entre componentes e as responsabilidades funcionais associadas a cada entidade do sistema.

Em particular, a aplicação consome os seguintes serviços:

- **VoterRegistrationService / IssueVotingCredential** (VoterRequest → VoterResponse)
    - VoterRequest.citizen_card_number
    - VoterResponse.isEligible
    - VoterResponse.votingCredential
- **VotingService / GetCandidates** (GetCandidatesRequest → GetCandidatesResponse)
    - devolve candidates[], cada elemento com {id, name}
- **VotingService / Vote** (VoteRequest → VoteResponse)
    - VoteRequest.votingCredential
    - VoteRequest.candidateId
    - VoteResponse.success
    - VoteResponse.message
- **VotingService / GetResults** (GetResultsRequest → GetResultsResponse)
    - devolve results[], com {id, name, votes}

Esta separação contratual reflete de forma clara o particionamento conceptual discutido na **Tarefa 1**, distinguindo explicitamente a fase de **registo e credenciação de eleitores** (associada à AR) das fases de

votação e apuramento de resultados (associadas à AV). Tal distinção contribui para a preservação do anonimato do voto, reforçando a separação de responsabilidades entre entidades e reduzindo a possibilidade de correlação entre identidade e opção de voto.

4.2. Arquitetura lógica da aplicação

A aplicação desenvolvida assume a forma de uma aplicação web composta por duas camadas principais: (i) uma **interface de utilizador** (frontend), e (ii) uma **camada de orquestração** (backend), responsável por coordenar o processo de votação e consumir os serviços gRPC remotos disponibilizados em ken01.utad.pt:9091.

Importa salientar que a aplicação **não implementa serviços gRPC próprios**. O seu papel limita-se a consumir os serviços existentes, coordenar o processo de votação em três fases e gerir o estado mínimo necessário ao nível da sessão do utilizador. Esta opção reduz a complexidade da solução e está alinhada com os objetivos pedagógicos da unidade curricular.

A arquitetura lógica inclui os seguintes componentes mínimos:

1. **Interface Web (Web UI)**, organizada em três vistas principais:
    - Registo;
    - Votação;
    - Apuramento de resultados.
2. **Camada de orquestração (cliente gRPC)**:
    - módulo RegistrationClient, responsável pela invocação do serviço IssueVotingCredential;
    - módulo VotingClient, responsável pela invocação dos serviços GetCandidates, Vote e GetResults.
3. **Gestão de estado local (sessão)**:
    - armazenamento temporário da votingCredential atribuída ao utilizador;
    - manutenção de um marcador lógico de voto efetuado, utilizado para impedir múltiplas submissões de voto na própria aplicação.

Nota crítica de robustez:

Conforme evidenciado na **Tarefa 2**, o serviço de votação fornecido (mock) aceita a reutilização da mesma credencial de voto. Para mitigar este comportamento no contexto do protótipo, a aplicação impõe uma restrição local de "uma submissão por credencial", preservando a coerência funcional do processo e aproximando o comportamento observado daquele que seria esperado num sistema real.

4.3. Implementação do processo de votação em três fases

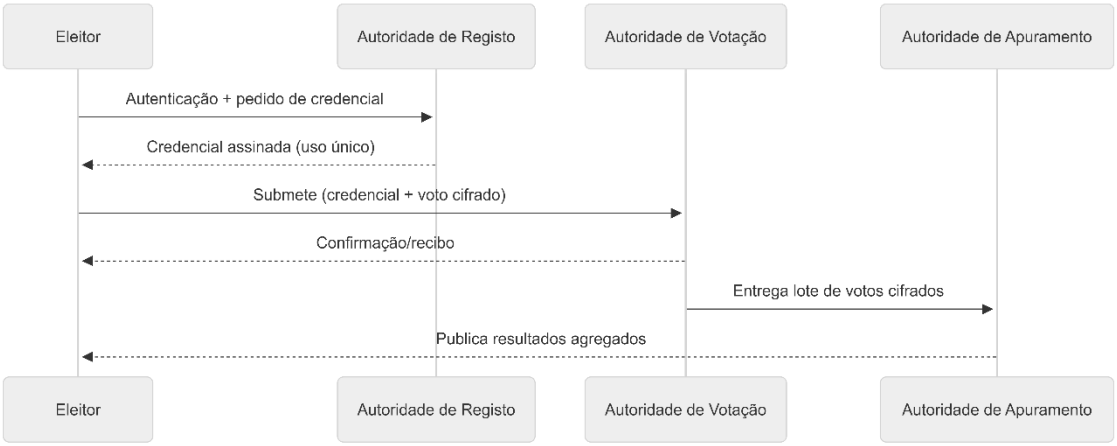

Figura 7 - Fases do processo de votação

4.3.1 Registo

Objetivo:
Recolher o número do Cartão de Cidadão do eleitor e obter uma credencial de votação junto da AR, sem criar qualquer associação posterior entre identidade e voto.

Passos na aplicação:

1. A interface solicita o valor citizen_card_number.
2. A aplicação invoca o serviço IssueVotingCredential.
3. Se isEligible = true, a votingCredential é armazenada temporariamente na sessão e o utilizador é encaminhado para a Fase 2.
4. Se isEligible = false, a aplicação informa o utilizador e impede o avanço no processo.

Garantia pretendida:
A identidade do eleitor é utilizada exclusivamente nesta fase para efeitos de validação e emissão da credencial. A partir deste momento, o processo de votação decorre sem reutilização de dados identificativos.

4.3.2 Votação

Objetivo:
Permitir ao eleitor submeter o seu voto utilizando apenas a credencial de votação, sem qualquer referência à sua identidade.

Passos na aplicação:

1. A aplicação invoca GetCandidates e apresenta a lista de candidatos.
2. O utilizador seleciona o candidato pretendido, identificado pelo respetivo id.
3. A aplicação invoca o serviço Vote com os parâmetros {votingCredential, candidateId}.

4. Se success = true, a aplicação regista localmente a credencial como utilizada e apresenta a confirmação de voto registado.
5. Em caso de falha, a aplicação apresenta a mensagem de erro devolvida pelo serviço.

Mitigação de robustez (com base na Tarefa 2):

- se a credencial já tiver sido utilizada na aplicação, novas tentativas de submissão são bloqueadas, mesmo que o serviço remoto aceite pedidos repetidos.

4.3.3 Apuramento

Objetivo:
Obter e apresentar os resultados agregados da votação, sem possibilidade de reidentificação dos eleitores.

Passos na aplicação:

1. A aplicação invoca o serviço GetResults.
2. Os resultados são apresentados sob a forma de uma tabela contendo {candidato, número de votos}.
3. Opcionalmente, é disponibilizada a funcionalidade de atualização dos resultados.

4.4 Justificação da escolha da tecnologia e framework

A aplicação desenvolvida assenta numa arquitetura orientada a serviços, recorrendo a tecnologias amplamente reconhecidas no domínio da Integração de Sistemas, nomeadamente **gRPC**, **FastAPI** e uma interface web estática baseada em **HTML, CSS e JavaScript**.

A escolha do **gRPC** fundamenta-se na sua adequação a cenários de comunicação eficiente e estruturada entre serviços distribuídos, através de contratos fortemente tipados definidos em ficheiros .proto. Esta abordagem promove interoperabilidade, validação precoce de erros e uma separação clara de responsabilidades entre consumidores e produtores de serviços, características particularmente relevantes em sistemas institucionais e em contextos da Administração Pública.

O **FastAPI** foi selecionado como framework para o backend de orquestração devido à sua simplicidade, elevado desempenho e suporte nativo para APIs REST modernas. Esta framework permite expor endpoints claros para consumo pelo frontend, funcionando como uma camada intermédia que coordena o processo de votação e encapsula a complexidade da interação com os serviços gRPC remotos. A sua integração natural com Python revelou-se adequada para um protótipo académico, permitindo simultaneamente a adoção de boas práticas de engenharia de software.

O **frontend estático** foi intencionalmente mantido simples, evitando dependências excessivas de frameworks de interface. Esta decisão reforça o caráter pedagógico do protótipo, permitindo concentrar a análise na integração de sistemas e no fluxo funcional do processo de votação, em vez da complexidade da camada de apresentação.

4.5 Casos de uso da aplicação

A aplicação suporta um conjunto reduzido, mas representativo, de casos de uso, diretamente alinhados com o processo de votação definido no enunciado.

UC1 — Registar eleitor e emitir credencial de votação

Este caso de uso permite ao eleitor iniciar o processo de votação, introduzindo o número do Cartão de Cidadão para validação de elegibilidade. A aplicação solicita ao serviço gRPC de registo a emissão de uma credencial de votação, a qual é devolvida apenas aos eleitores considerados elegíveis. A identidade do eleitor é utilizada exclusivamente nesta fase, não sendo reutilizada posteriormente.

UC2 — Consultar lista de candidatos

Neste caso de uso, o eleitor solicita a lista de candidatos disponíveis para votação. A aplicação consome o serviço gRPC de votação para obter a informação atualizada, apresentando-a de forma legível no frontend. Este caso de uso é independente da identidade do eleitor, dependendo apenas da disponibilidade do serviço.

UC3 — Submeter voto

O eleitor seleciona um candidato e submete o seu voto utilizando exclusivamente a credencial de votação previamente emitida. A aplicação valida localmente o estado da sessão e encaminha o pedido para o serviço gRPC de votação. Em caso de sucesso, o voto é registado e a credencial é marcada como utilizada no contexto da aplicação.

UC4 — Consultar resultados da votação

Este caso de uso permite consultar os resultados agregados da votação. A aplicação solicita ao serviço gRPC de apuramento os totais por candidato e apresenta essa informação ao utilizador, sem qualquer associação a eleitores individuais.

4.5.1 Diagrama de casos de uso

O diagrama de casos de uso reflete os principais atores e funcionalidades da aplicação. O ator principal é o **Eleitor**, que interage com o sistema para realizar o processo de votação. Os casos de uso identificados são:

- Registar eleitor e emitir credencial;
- Consultar candidatos;
- Submeter voto;
- Consultar resultados.

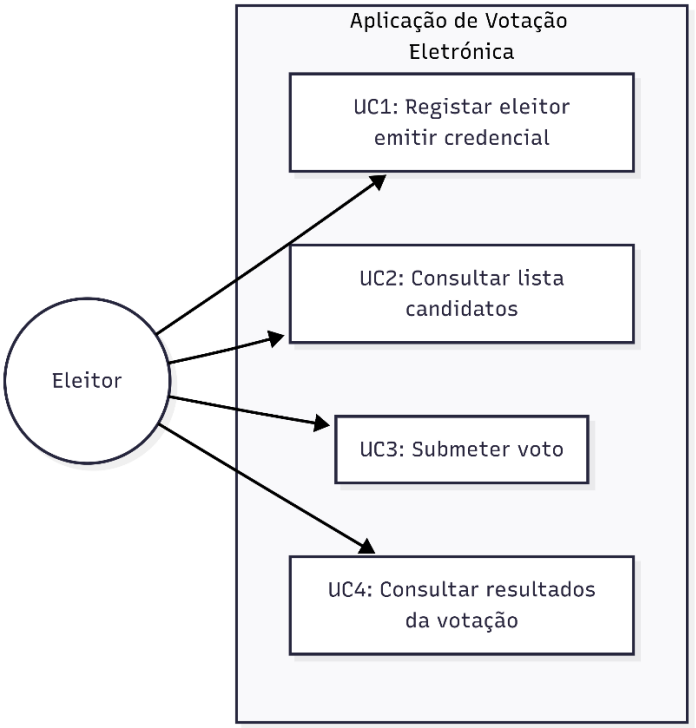

Figura 8 - Diagrama de casos de uso da aplicação de votação eletrónica

O diagrama de casos de uso apresentado sintetiza as principais funcionalidades disponibilizadas pela aplicação de votação eletrónica. O ator principal é o Eleitor, que interage com o sistema para realizar o processo de votação ao longo das suas diferentes fases. Os casos de uso identificados correspondem ao registo e emissão de credencial, consulta da lista de candidatos, submissão do voto e consulta dos resultados da votação. Este diagrama reflete de forma direta e simplificada o fluxo funcional da aplicação, estando alinhado com os casos de uso descritos anteriormente e com o processo definido no enunciado do trabalho.

4.6 Interface gráfica associada aos casos de uso

Para cada caso de uso identificado, a aplicação disponibiliza uma interface gráfica correspondente, acessível através de uma única página web organizada em secções funcionais:

- **UC1** — Secção de registo e emissão de credencial;

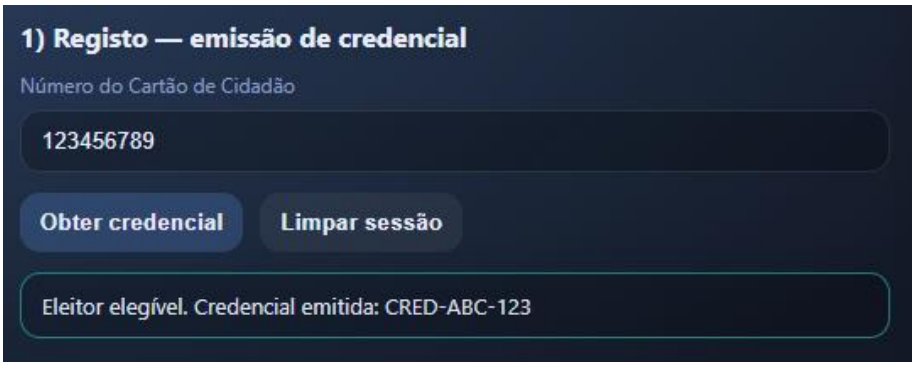

Figura 9 - Emissão de Credencial

- **UC2 e UC3** — Secção de votação (consulta de candidatos e submissão do voto);

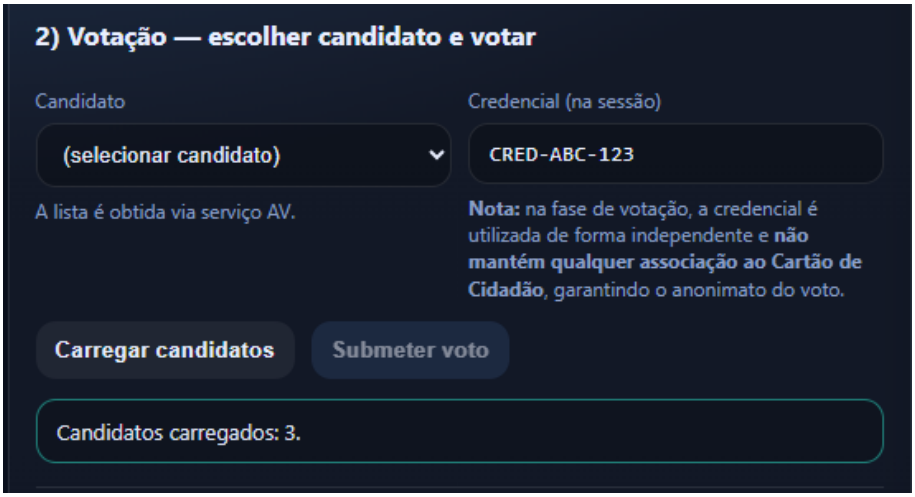

Figura 10 - Candidatos e Voto

- **UC4** — Secção de apuramento e apresentação de resultados.

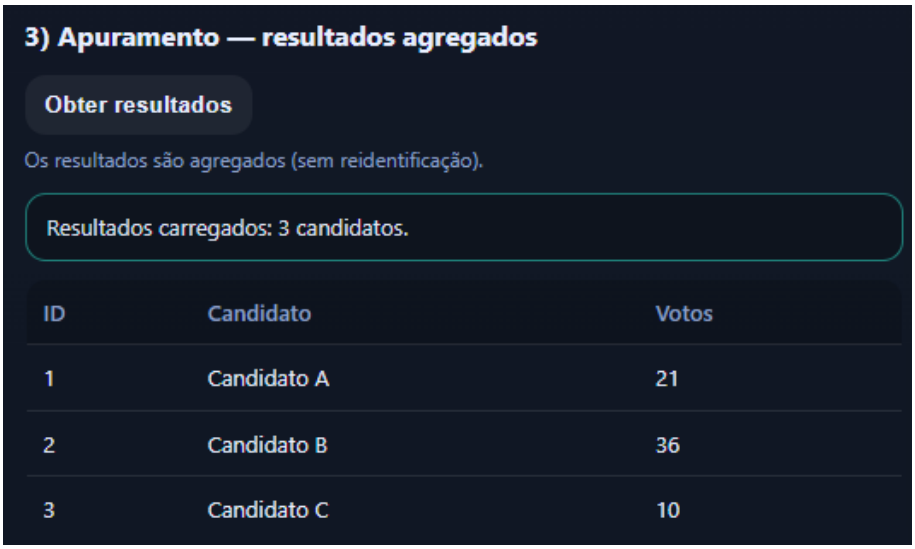

Figura 11 - Resultados

As imagens apresentadas correspondem a capturas de ecrã obtidas durante a execução do protótipo em ambiente local, garantindo a coerência entre o comportamento observado e a descrição funcional apresentada ao longo do relatório. A inclusão destas evidências visuais visa reforçar a transparência do processo de desenvolvimento e facilitar a validação dos casos de uso por parte do avaliador.

4.7 Processo de desenvolvimento e interação com ferramentas de IA

O desenvolvimento da aplicação seguiu uma abordagem incremental e iterativa, combinando implementação prática com reflexão crítica sobre as decisões tomadas. O processo iniciou-se com a análise dos contratos gRPC fornecidos, seguida do desenho de uma arquitetura simples que respeitasse os princípios de separação de responsabilidades e anonimato do voto.

A interação com o ChatGPT foi utilizada como ferramenta de apoio ao desenvolvimento, nomeadamente para:

- clarificação de conceitos arquiteturais;
- apoio à definição da estrutura da aplicação;

- identificação e análise de erros técnicos durante a implementação;
- revisão e melhoria do texto académico do relatório.

Importa salientar que o ChatGPT foi utilizado como instrumento de suporte e não como substituto do processo de decisão. As opções finais adotadas resultam de análise crítica, validação empírica através de testes e alinhamento com os objetivos pedagógicos da unidade curricular.

4.8 Repositório GitHub e instruções de execução

O código-fonte completo do projeto encontra-se disponível no seguinte repositório GitHub:

https://github.com/AndreMacielSousa/VotingSystem-App

O repositório pode ser clonado e executado localmente, sendo disponibilizadas instruções detalhadas para a criação do ambiente virtual, instalação das dependências, execução do backend, execução do frontend e realização dos testes funcionais, permitindo a reprodução integral dos resultados apresentados neste relatório.

4.9 Reflexão sobre possíveis evoluções da arquitetura

Embora o protótipo desenvolvido cumpra os objetivos propostos, a sua evolução para um sistema de votação em produção exigiria alterações significativas, nomeadamente a integração real com sistemas institucionais de identidade e recenseamento, a adoção de mecanismos criptográficos formais para proteção e verificação dos votos, o reforço dos contratos de erro e da idempotência, e a utilização de infraestruturas com suporte a alta disponibilidade, auditoria e governança.

4.10 Considerações técnicas sobre a integração gRPC

Durante a implementação, verificou-se que a utilização direta de um cliente gRPC em Python não era viável devido à configuração TLS do endpoint remoto, que apresentava um certificado “default” sem definição de *Subject Alternative Name*. Esta limitação impedia a validação de hostname exigida pelas bibliotecas gRPC modernas.

Para contornar esta restrição externa, foi adotada uma solução de mitigação baseada na utilização da ferramenta grpcurl, invocada pelo backend como subprocesso, recorrendo à opção -insecure. Esta abordagem permitiu manter comunicação cifrada e assegurar o correto funcionamento do protótipo, sem comprometer os objetivos funcionais da tarefa nem o fluxo definido no enunciado.

4.11 Validação e testes da aplicação desenvolvida

Após a definição da arquitetura e a implementação da aplicação de votação baseada em gRPC, procedeu-se à realização de um conjunto de testes funcionais em ambiente local, com o objetivo de validar o correto funcionamento do protótipo e a integração entre os seus diferentes componentes. Estes testes incidem sobre o backend de orquestração (FastAPI), o frontend web estático e a interação com os serviços gRPC remotos disponibilizados.

4.11.1 Disponibilidade do backend

Como primeiro passo, foi validada a correta execução do backend de orquestração através da invocação do endpoint de saúde (/health). Este endpoint permite confirmar que o servidor FastAPI se encontra ativo e apto a receber pedidos.

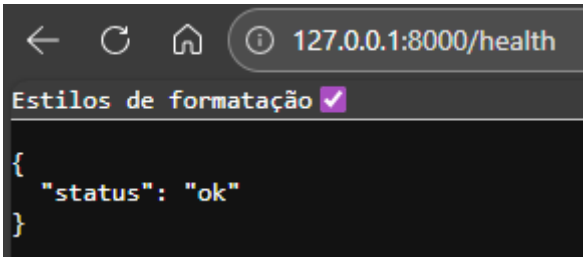

Figura 12 - Disponibilidade backend

A resposta obtida ({"status":"ok"}) confirma que o backend está operacional, constituindo uma condição prévia necessária para os testes subsequentes.

4.11.2 Registo e emissão de credencial

De seguida, testou-se o fluxo de registo de eleitores, correspondente à primeira fase do processo de votação. Neste teste, o utilizador introduz um número de Cartão de Cidadão (valor fictício, dado o carácter de mock do serviço) através da interface web, sendo o pedido encaminhado para o backend e, subsequentemente, para o serviço gRPC de registo.

```
PS C:\Users\Grupo EMAC\OneDrive - Universidade Aberta\Documents\GitHub> for ($i=1; $i -le 10; $i++) {
>>   $cc = (Get-Random -Minimum 100000000 -Maximum 999999999).ToString()
>>   $r = Invoke-WebRequest -Method POST http://127.0.0.1:8000/register `
>>         -Headers @{ "Content-Type"="application/json" } `
>>         -Body "{ `"citizen_card_number`": `"$cc`" }"
>>   $json = $r.Content | ConvertFrom-Json
>>   Write-Host "Tentativa $i | CC=$cc | is_eligible=$($json.is_eligible)"
>>   if ($json.is_eligible -eq $true -and $json.voting_credential) {
>>     Write-Host "CREDENCIAL: $($json.voting_credential)"
>>     break
>>   }
>> }
Tentativa 1 | CC=329820900 | is_eligible=True
CREDENCIAL: INVALID-577A25BFB899
PS C:\Users\Grupo EMAC\OneDrive - Universidade Aberta\Documents\GitHub> for ($i=1; $i -le 10; $i++) {
>>   $cc = (Get-Random -Minimum 100000000 -Maximum 999999999).ToString()
>>   $r = Invoke-WebRequest -Method POST http://127.0.0.1:8000/register `
>>         -Headers @{ "Content-Type"="application/json" } `
>>         -Body "{ `"citizen_card_number`": `"$cc`" }"
>>   $json = $r.Content | ConvertFrom-Json
>>   Write-Host "Tentativa $i | CC=$cc | is_eligible=$($json.is_eligible)"
>>   if ($json.is_eligible -eq $true -and $json.voting_credential) {
>>     Write-Host "CREDENCIAL: $($json.voting_credential)"
>>     break
>>   }
>> }
Tentativa 1 | CC=456797048 | is_eligible=True
CREDENCIAL: CRED-DEF-456
```

Figura 13 - Obter Credencial de Votação

Quando o serviço indica que o eleitor é elegível (is_eligible = true), é devolvida uma credencial de votação, a qual é armazenada temporariamente na sessão do frontend. Este teste confirma que:

- o frontend comunica corretamente com o backend;
- o backend consome o serviço gRPC de registo;
- a credencial é emitida sem persistir qualquer associação entre identidade e voto futuro.

4.11.3 Votação com credencial válida

Na segunda fase, foi testado o processo de votação propriamente dito. A aplicação solicita a lista de candidatos através do serviço gRPC de votação, apresenta essa lista ao utilizador e permite a submissão do voto utilizando exclusivamente a credencial previamente obtida.

```
PS C:\Users\Grupo EMAC\OneDrive - Universidade Aberta\Documents\GitHub> curl -Method POST http://127.0.0.1:8000/vote `
>>   -Headers @{ "Content-Type"="application/json" } `
>>   -Body '{ "voting_credential": "CRED-DEF-456", "candidate_id": 3 }'


StatusCode        : 200
StatusDescription : OK
Content           : {"success":true,"message":"Voto registado para o Candidato C."}
RawContent        : HTTP/1.1 200 OK
                    Content-Length: 63
                    Content-Type: application/json
                    Date: Sun, 01 Feb 2026 11:15:48 GMT
                    Server: uvicorn

                    {"success":true,"message":"Voto registado para o Candidato C."}
Forms             : {}
Headers           : {[Content-Length, 63], [Content-Type, application/json], [Date, Sun, 01 Feb 2026 11:15:48 GMT], [Server, uvicorn]}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 63
```

Figura 14 - Voto com Credencial Válida

A resposta positiva do serviço (success = true) confirma que o voto foi aceite e registado. Este teste demonstra que a fase de votação decorre sem reutilização de dados identificativos, cumprindo o princípio do anonimato do voto.

4.11.4 Mitigação da reutilização de credenciais

Conforme identificado na Tarefa 2, os serviços gRPC disponibilizados aceitam, em contexto de mock, a reutilização da mesma credencial de voto. Para mitigar este comportamento no protótipo desenvolvido, foi implementado um mecanismo de bloqueio local na aplicação.

Neste teste, tentou-se submeter novamente um voto utilizando a mesma credencial já usada anteriormente.

```
PS C:\Users\Grupo EMAC\OneDrive - Universidade Aberta\Documents\GitHub> curl -Method POST http://127.0.0.1:8000/vote `
>>   -Headers @{ "Content-Type"="application/json" } `
>>   -Body '{ "voting_credential": "CRED-ABC-123", "candidate_id": 1 }'
curl : {"detail":"Esta credencial já foi usada nesta aplicação (bloqueio local do protótipo)."}
At line:1 char:1
+ curl -Method POST http://127.0.0.1:8000/vote `
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidOperation: (System.Net.HttpWebRequest:HttpWebRequest) [Invoke-WebRequest], WebException
    + FullyQualifiedErrorId : WebCmdletWebResponseException,Microsoft.PowerShell.Commands.InvokeWebRequestCommand
```

Figura 15 - Mitigação Credencial Reutilizada

A rejeição da operação confirma a eficácia da mitigação local implementada, preservando a coerência funcional do processo e aproximando o comportamento da aplicação ao esperado num sistema de votação real.

4.11.5 Apuramento de resultados

Por fim, testou-se a fase de apuramento, na qual a aplicação solicita os resultados agregados ao serviço gRPC correspondente e apresenta essa informação ao utilizador.

```
PS C:\Users\Grupo EMAC\OneDrive - Universidade Aberta\Documents\GitHub> curl http://127.0.0.1:8000/results


StatusCode        : 200
StatusDescription : OK
Content           : {"results":[{"id":1,"name":"Candidato A","votes":21},{"id":2,"name":"Candidato B","votes":36},{"id":3,"name":"Candidato C","votes":11}]}
RawContent        : HTTP/1.1 200 OK
                    Content-Length: 136
                    Content-Type: application/json
                    Date: Sun, 01 Feb 2026 11:18:09 GMT
                    Server: uvicorn

                    {"results":[{"id":1,"name":"Candidato A","votes":21},{"id":2,"name":"Candi...
Forms             : {}
Headers           : {[Content-Length, 136], [Content-Type, application/json], [Date, Sun, 01 Feb 2026 11:18:09 GMT], [Server, uvicorn]}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 136
```

Figura 16 - Apuramento Resultados

Os resultados apresentados incluem apenas contagens globais por candidato, não existindo qualquer possibilidade de reidentificação dos eleitores. Este teste valida o correto funcionamento da terceira fase do processo de votação e a separação entre voto individual e apuramento global.

5. Discussão Integrada da Solução Proposta

A análise integrada do trabalho desenvolvido ao longo das três tarefas permite avaliar de forma crítica a coerência da solução proposta, bem como a adequação das decisões arquiteturais, tecnológicas e de integração adotadas face aos objetivos da unidade curricular de Integração de Sistemas. Esta discussão articula a proposta conceptual da arquitetura (Tarefa 1), a análise empírica dos serviços gRPC disponibilizados (Tarefa 2) e a implementação concreta da aplicação de votação eletrónica (Tarefa 3), evidenciando a relação entre desenho, limitações dos serviços e opções de implementação.

A arquitetura definida na Tarefa 1 estabelece, desde o início, um conjunto claro de princípios estruturantes, nomeadamente a separação de responsabilidades, a distribuição de confiança e a minimização de dados sensíveis. A divisão do sistema em Autoridade de Registo, Autoridade de Votação e Autoridade de Apuramento revelou-se fundamental para garantir, pelo menos ao nível conceptual, a conciliação entre validação rigorosa dos votantes e anonimato do voto. Estes princípios encontram suporte na literatura sobre arquiteturas orientadas a serviços e integração de sistemas, sendo particularmente relevantes em contextos institucionais e de Administração Pública, onde a governação e a transparência assumem um papel central.

Os resultados obtidos na Tarefa 2 evidenciaram, contudo, um conjunto de limitações relevantes nos serviços gRPC fornecidos em contexto de *mock*, que condicionam diretamente a robustez de qualquer aplicação que os consuma. Entre estas limitações destacam-se a inexistência de controlo efetivo da unicidade da credencial de voto, a fragilidade dos contratos de erro, a ausência de garantias explícitas de idempotência e consistência distribuída, bem como a inexistência de mecanismos de observabilidade, auditoria e governação. Estas limitações são típicas de serviços simplificados para fins pedagógicos, mas ilustram de forma clara os desafios reais enfrentados em cenários de integração de sistemas distribuídos.

A aplicação desenvolvida na Tarefa 3 pode ser interpretada como uma resposta direta e consciente a essas limitações. As decisões de desenho da aplicação não se limitaram a consumir passivamente os serviços disponíveis, mas procuraram mitigar, sempre que possível, fragilidades identificadas na fase de testes. Um exemplo particularmente relevante é a implementação de um mecanismo local de bloqueio da reutilização de credenciais de voto, que, embora não substitua um controlo transacional ao nível do serviço, demonstra a capacidade de introduzir coerência funcional na camada de orquestração. Esta

decisão evidencia uma compreensão clara da distinção entre responsabilidades do serviço e responsabilidades da aplicação cliente, aspeto central na engenharia de integração de sistemas.

De igual modo, a introdução de uma camada intermédia baseada em FastAPI, responsável pela orquestração do processo de votação, permitiu encapsular a complexidade da interação com os serviços gRPC e expor uma interface mais simples ao frontend. Esta opção reforça o desacoplamento entre camadas, facilita a validação do fluxo funcional e aproxima a solução desenvolvida de padrões arquiteturais comuns em ambientes reais, onde a orquestração de serviços raramente é feita diretamente na camada de apresentação.

No que respeita à integração técnica com os serviços gRPC, a necessidade de recorrer à ferramenta *grpcurl* como mecanismo de mitigação para limitações de configuração TLS do endpoint remoto constitui um exemplo claro de constrangimento externo ao desenho da aplicação. Longe de representar uma fragilidade conceptual da solução, esta situação ilustra a importância de lidar com ambientes heterogéneos e imperfeitos, uma realidade frequente em projetos de integração de sistemas. A abordagem adotada permitiu manter o foco nos objetivos pedagógicos do trabalho, sem comprometer a análise do processo de votação nem a validação do fluxo funcional definido no enunciado.

As limitações associadas à demonstração online da aplicação devem igualmente ser interpretadas à luz desta distinção entre solução conceptual e contexto de execução. As restrições impostas pelo ambiente de alojamento — nomeadamente a inexistência de suporte para processos persistentes e as limitações de segurança associadas a *mixed content* — não invalidam os resultados obtidos, uma vez que a execução completa e validada do protótipo ocorre em ambiente local. Esta separação entre demonstração visual e execução funcional reforça a importância de avaliar soluções de integração com base no seu desenho e comportamento técnico, e não exclusivamente nas condições do ambiente onde são apresentadas.

Em síntese, a solução desenvolvida evidencia uma coerência consistente entre arquitetura, análise dos serviços e implementação da aplicação. As decisões tomadas refletem uma compreensão clara dos princípios de Integração de Sistemas, nomeadamente no que diz respeito à separação de responsabilidades, à definição de contratos, à gestão de limitações externas e à necessidade de governação e validação empírica. Embora a solução esteja limitada pelo caráter simplificado dos serviços disponibilizados e pelo contexto académico do trabalho, os resultados obtidos demonstram a capacidade de conceber, analisar e implementar uma solução integrada de forma crítica e fundamentada.

5.1 Limitações do ambiente de execução e implicações na demonstração

A aplicação desenvolvida encontra-se parcialmente publicada em ambiente online, através do endereço https://andremaciel.pt/IS2026, disponibilizando uma interface web de carácter demonstrativo. Contudo, o backend da aplicação não é exposto publicamente neste servidor.

Esta limitação decorre de constrangimentos técnicos do alojamento utilizado, nomeadamente:

- inexistência de suporte para processos persistentes no servidor;
- ausência de mecanismos configuráveis de reverse proxy;
- restrições de segurança dos browsers relativamente a chamadas HTTP a partir de páginas servidas em HTTPS (*mixed content*).

Deste modo, a execução funcional completa do protótipo — envolvendo frontend, backend e consumo dos serviços gRPC remotos — é realizada e validada em ambiente local. Esta opção não compromete

os objetivos do trabalho, uma vez que o foco reside na integração de sistemas, na validação do processo de votação e na análise crítica das limitações identificadas.

As instruções detalhadas para a execução local da aplicação encontram-se documentadas no repositório GitHub do projeto, permitindo ao avaliador reproduzir integralmente os testes apresentados neste relatório.

6. Conclusão

O trabalho desenvolvido no âmbito da unidade curricular de Integração de Sistemas teve como objetivo a análise, conceção e implementação de um protótipo académico de aplicação de votação eletrónica, explorando de forma prática os desafios associados à integração de serviços distribuídos, à definição de contratos e à robustez de soluções orientadas a serviços.

A proposta arquitetural apresentada demonstrou que a separação clara de responsabilidades entre entidades funcionais — Autoridade de Registo, Autoridade de Votação e Autoridade de Apuramento — constitui um princípio fundamental para conciliar validação dos votantes e anonimato do voto. Esta abordagem, alinhada com princípios clássicos de integração de sistemas e arquiteturas orientadas a serviços, revelou-se adequada ao contexto académico e permitiu enquadrar de forma consistente o papel dos sistemas da Administração Pública, sem comprometer a minimização de dados sensíveis.

A análise dos serviços gRPC disponibilizados evidenciou limitações estruturais relevantes, típicas de serviços mock, mas particularmente elucidativas do ponto de vista pedagógico. A inexistência de garantias de unicidade do voto, a fragilidade dos contratos de erro, a ausência de mecanismos de idempotência, auditoria e governação, bem como as limitações de segurança observadas nos testes, reforçaram a importância de contratos bem definidos e de práticas de governação em cenários reais de integração de sistemas distribuídos.

A aplicação desenvolvida procurou responder de forma consciente a essas limitações, não apenas consumindo os serviços disponíveis, mas introduzindo mecanismos de mitigação na camada de orquestração sempre que tal se revelou necessário. As decisões tomadas — nomeadamente a introdução de controlo local de reutilização de credenciais, a utilização de uma camada intermédia de orquestração baseada em FastAPI e a validação empírica do fluxo de votação em ambiente local — evidenciam uma compreensão clara da distinção entre responsabilidades dos serviços e responsabilidades da aplicação cliente.

Importa sublinhar que as limitações observadas na demonstração online e na integração técnica com os serviços gRPC decorrem maioritariamente de constrangimentos externos ao desenho da solução, não comprometendo os objetivos fundamentais do trabalho. Pelo contrário, estas limitações contribuíram para reforçar a análise crítica do processo de integração e para evidenciar a importância de adaptar soluções a contextos técnicos imperfeitos, uma realidade frequente em projetos de integração de sistemas.

Em síntese, o trabalho permitiu consolidar conhecimentos teóricos e práticos sobre Integração de Sistemas, demonstrando a capacidade de analisar criticamente serviços distribuídos, conceber arquiteturas coerentes e implementar soluções funcionais alinhadas com requisitos funcionais e não funcionais exigentes. Embora a solução apresentada não se destine a um contexto de produção, constitui uma base sólida para compreender os desafios reais associados ao desenho e integração de sistemas complexos, bem como para identificar caminhos de evolução futura em cenários institucionais mais exigentes.

Referências Bibliográficas

FIELDING, Roy Thomas. *Architectural styles and the design of network-based software architectures*. 2000. Tese (Doutoramento em Ciência da Computação) – University of California, Irvine, Irvine, 2000.

FASTAPI. *FastAPI Documentation*. [S. l.], s. d. Disponível em: https://fastapi.tiangolo.com. Acesso em: 27 jan. 2026.

GOOGLE. *Protocol Buffers: JSON Mapping*. [S. l.], s. d. Disponível em: https://protobuf.dev/programming-guides/proto3/#json. Acesso em: 14 jan. 2026.

GRPCURL. *gRPCurl: A command-line tool for interacting with gRPC servers*. [S. l.], s. d. Disponível em: https://github.com/fullstorydev/grpcurl. Acesso em: 21 jan. 2026.

MAHMOOD, Zaigham. *Service-oriented architecture: potential benefits and challenges*. In: WSEAS International Conference on Computers, 11., 2007, Agios Nikolaos. **Proceedings** […]. Agios Nikolaos: WSEAS Press, 2007.

MDN WEB DOCS. *Mixed content*. [S. l.], s. d. Disponível em: https://developer.mozilla.org/en-US/docs/Web/Security/Mixed_content. Acesso em: 19 jan. 2026.

NIKNEJAD, Naghmeh et al. *Understanding service-oriented architecture (SOA): a systematic literature review and directions for further investigation*. Information Systems, v. 91, p. 101491, 2020.

RUH, William A.; MAGINNIS, Francis X.; BROWN, William J. *Enterprise application integration: a Wiley tech brief*. New York: John Wiley & Sons, 2001.

UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO. *Voting system: enunciado e especificações do processo e dos serviços gRPC*. Vila Real: UTAD, 2025.