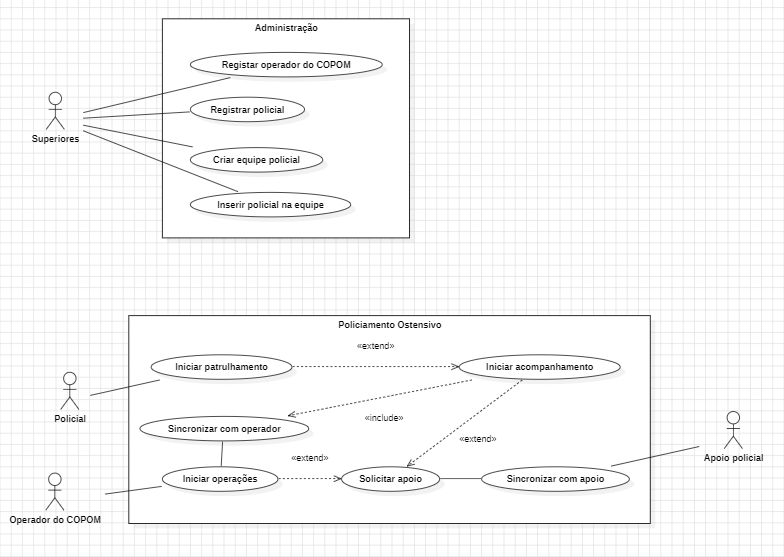
# PROCEDIMENTOS METODOLÓGICOS

## Visão geral do sistema

Na figura 29 abaixo está um diagrama de caso de uso representando o planejamento geral do sistema, com seus atores e principais interações:

Figura 29 - Diagrama de caso de uso representando o planejamento geral do sistema



Fonte: Própria, 2023

Nesse cenário, existem quatro atores (*stakeholders*) que têm interesse direto no projeto e que podem ser afetados por suas atividades ou resultados:

* Policial: agentes que trabalham no campo (ruas), geralmente em duplas, como no caso dos motos patrulheiros, ou em equipes, como os policiais que trabalham em viatura quatro rodas; nesse caso de uso, eles são os agentes responsáveis por terem iniciado uma ocorrência de acompanhamento;
* Operador do COPOM: agente responsável pelo por recebimento de uma ocorrência de acompanhamento, para auxiliar o policial em campo na tomada de decisão;
* Apoio policial: agentes que estendem as características de um policial; no entanto, estes não iniciaram uma ocorrência, mas, foram solicitados a prestar apoio em uma;
* Superiores: são os líderes dos agentes, como por exemplo sargentos, subtenentes e tentes; são responsáveis por gerenciar equipes policias.

Também, esse caso de uso tem dois *subjects* (divisões do sistema): administração e policiamento ostensivo.

Na administração, somente os superiores terão acesso, nele temos alguns casos de uso responsáveis por fazer o famoso *create*, *read*, *update* e *delete* (CRUD). A partir daí, cadastros são gerados para que os outros atores possam usar o sistema. Tudo isso, para evitar acesso público ao sistema e uma melhor gestão de pessoas.

No policiamento ostensivo é onde resolvemos a hipótese de trabalho: um sistema de tempo real com interface gráfica para resolver operações de cerco e acompanhamento da PM. Os policiais de campo e os operadores do COPOM iniciam (caso de uso) suas jornadas de trabalho, se colocando à disposição no sistema para lidar com ocorrências. Os policiais podem ou não iniciar um acompanhamento. Caso aconteça, ele deve ser sincronizado com um operador do COPOM – é reponsabilidade do sistema garantir essa sincronização. Durante o acompanhamento, o policial pode ou não solicitar apoio caso sinta a necessidade. O operador do COPOM, auxiliando um policial no acompanhamento, também pode solicitar apoio caso sinta a necessidade.

## Metodologia de gestão

Para a gestão do desenvolvimento do sistema foi escolhido como metodologia o *Scrum*. Essa metodologia permite entregar muito valor em pouco tempo, pois prioriza os casos de uso de maior valor para o sistema; aqueles itens que trarão mais valor para os *stakeholders*.

Além disso, essa metodologia tem como pilar a inspeção e adaptação contínua, o que consegue extrair o melhor da equipe de desenvolvimento, impedido que trabalho desnecessário seja realizado e que barreiras sejam removidas. Também, a transparência é um pilar que ajuda a entender o ponto em que se encontra o desenvolvimento e se um objetivo será alcançado ou não.

Por fim, as reuniões de planejamento são muito melhores por enfrentar a realidade. Nas palavras do criador do Scrum:

[...] próprio ato de planejar é tão sedutor, tão atraente, que o planejamento em si se torna mais importante do que o plano. E o plano se torna mais importante do que a realidade. Nunca se esqueça disto: o mapa não é o terreno (SUTHERLAND, 2019, p. 132).

Nesse modelo, a equipe é composta pelas seguintes pessoas:

Tabela 4 - Equipe de desenvolvimento

|  |  |
| --- | --- |
| Equipe de desenvolvimento | |
| Integrante | Reponsabilidade |
| Marcos Paulo F. Vaz | *Scrum Master* |
| Matheus dos Santos Silva | Desenvolvedor |
| Matheus Oliveira de Moraes | Desenvolvedor |
| Samuel Araujo de Souza | Desenvolvedor, *Product Owner* |

Fonte: Própria, 2023

## Requisitos do sistema

Tendo em vista a visão geral do sistema, o seguinte *backlog* foi gerado:

Tabela 5 - Backlog inicial com as histórias de usuário da visão geral do sistema

|  |  |  |
| --- | --- | --- |
| Título | Descrição | Tipo |
| Iniciar patrulhamento | Como policial, quero iniciar um patrulhamento para iniciar minha jornada de trabalho e ser capaz de iniciar um acompanhamento ou prestar apoio quando necessário | User story |
| Iniciar acompanhamento | Como policial, quero iniciar um acompanhamento para que minha localização seja transmitida automaticamente e eu evite a modulação manual | User story |
| Solicitar apoio | Como policial, durante um acompanhamento, quero solicitar apoio, caso seja necessário, para realização do cerco no indivíduo em fuga | User story |
| Visualizar perseguidor | Como policial, no apoio, quero ser visualizar a posição do policial que solicitou ajuda para ser capaz de planejar o deslocamento para o cerco | User story |
| Iniciar operações | Como operador do COPOM, quero iniciar as operações para iniciar minha jornada de trabalho e ser capaz de auxiliar policiais em acompanhamento | User story |
| Monitorar posições | Como operador do COPOM, quero visualizar a posição de todos policiais em patrulhamento para fazer uma melhor gestão do efetivo policial | User story |
| Módulo GPS | Analisar qual o melhor módulo GPS e como fazer integração com o policial de campo e seu celular | Tarefa |
| Gestão de equipes | Como supervisor, quero a criar policiais, suas equipes e operadores do COPOM para gerenciá-los | User story |
| Design system | Criar todo o design do sistema no Figma | Tarefa |
| Mostrar endereço | Como policial, no acompanhamento, quero saber o endereço em que me encontro para, se necessário, transmitir verbalmente | User story |
| Mostrar “rastro” | Como operador do COPOM, quero visualizar o rastro de um policial em acompanhamento para que possamos entender o padrão de fuga dos indivíduos | User story |
| Canal de voz | Como policial, em acompanhamento, quero poder iniciar um canal de voz, se necessário, com operador do COPOM para que possamos trocar informações verbalmente | User story |
| Transmissão ao vivo | Como operador do COPOM, quero acompanhar uma *livestream* do policial em acompanhamento para capturar informações do indivíduo em fuga (placa, veículo, cores, etc) | User story |
| Feed de notícias | Como policial, em patrulhamento, quero um feed de notícias para visualizar denúncias, veículos roubados e outras informações das redondezas | User story |

Fonte: Própria, 2023

Os itens no topo do *backlog* são os mais prioritários. À medida que se desce na lista a prioridade vai diminuindo. Essa priorização serve para evitar o pensamento de que tudo é importante. De acordo com Jeff Sutherland (2019, grifo nosso):

A pergunta que precisam fazer é: o que agregará mais valor ao projeto? Faça isso primeiro. No desenvolvimento de softwares, a regra, criada a partir de décadas, é que 80% do valor de qualquer programa estão em 205 de suas funcionalidades. Pense nisto: quando foi a última vez que você usou o editor do Visual Basic no Microsoft Word? Você provavelmente não sabe o que é Visual Basic, muito menos por que precisaria usar essa ferramenta. Mas ela está lá, e alguém gastou tempo em implementá-la, mas garanto que ela não aumenta o valor agregado do Word de maneira significativa.

No entanto, essa priorização não é fixa. Os itens podem se inspecionados e adaptados a cada *sprint* de acordo com o feedback do usuário ou uma nova necessidade percebida. Além disso, novos itens podem entrar ou ser excluídos do backlog. Itens podem sofrer mudança na sua descrição; podem ser subdivididos em itens menores; podem ser mesclados, etc. Por que disso? Segundo Jeff Sutherland (2014):

Na essência, o Scrum se baseia em uma ideia simples: quando começamos um projeto, por que não verificar a intervalos regulares se ele está indo no caminho certo e se isso é de fato o que as pessoas querem? E por que não se perguntar se é possível aprimorar a forma como você está trabalhando para obter resultados melhores e mais rápidos, e o que estaria impedindo você de fazer isso? O nome disso é ciclo de “inspeção e adaptação”. De tempos em tempos, pare o que está fazendo, revise o que já fez e verifique se deveria continuar fazendo a mesma coisa e como poderia fazer melhor.

O nome de todo esse processo é *grooming*, ou refinamento de backlog, em português, e é responsabilidade do dono do produto manter o escopo atualizado.

### Requisitos funcionais

Aqui segue-se uma sequência mais detalhada dos requisitos funcionais (caso de uso) e não funcionais (qualidade) do sistema. Por convenção, a referência a requisitos é feita através do identificador do requisito. Por exemplo, o requisito [RF016] é identificado pelo RF (Requisito Funcional) e do número 016. Já o requisito não funcional [NF008] é identificado pelo NF (Não Funcional) e por 008.

Para estabelecer a prioridade dos requisitos foram adotadas as denominações “essencial”, “importante” e “desejável”.

* **Essencial** é o requisito sem o qual o sistema não entra em funcionamento. Requisitos essenciais são requisitos imprescindíveis, que têm que ser implementados impreterivelmente;
* **Importante** é o requisito sem o qual o sistema entra em funcionamento, mas de forma não satisfatória. Requisitos importantes devem ser implementados, mas, se não forem, o sistema poderá ser implantado e usado mesmo assim;
* **Desejável** é o requisito que não compromete as funcionalidades básicas do sistema, isto é, o sistema pode funcionar de forma satisfatória sem ele. Requisitos desejáveis são requisitos que podem ser deixados para versões posteriores do sistema, caso não haja tempo hábil para implementá-los na versão que está sendo especificada.

#### [RF001] Cadastrar equipe de campo

Este caso de uso descreve como a um gerente pode cadastrar equipes de policiais em campo. Apesar da individualidade de cada soldado, as ocorrências envolvem uma equipe cadastrada.

**Ator:** Superiores dos soldados.

**Prioridade:** Essencial.

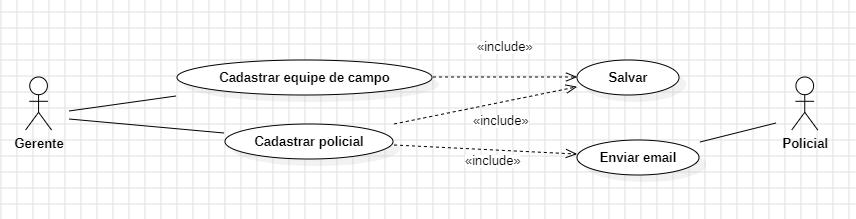
**Pré-condições:** O ator deve estar autenticado no sistema e ter a devida autorização para cadastrar as equipes de policiais.

**Pós-condições**: Equipe cadastrada; policiais individuais com acesso ao sistema; nova equipe listada na dashboard de controle.

**Fluxo principal**:

1. O superior das equipes (*admin*) clica no botão “Cadastrar Nova Equipe” numa *dashboard* de controle;
2. O sistema exibe o formulário de criação de equipes;
3. O admin preenche os campos obrigatório para cada membro;
4. O sistema valida os dados em tempo real;
5. O admin clica em “Cadastrar”;
6. O sistema envia um e-mail para cada policial cadastrado com seu usuário e senha.

Figura 30 - Caso de uso para cadastros de equipes de policiais



Fonte: Autoria própria, 2023

#### [RF002] Cadastrar operador do COPOM

Este caso de uso descreve como a um gerente pode cadastrar um operador do COPOM.

**Ator:** Superiores dos soldados.

**Prioridade:** Essencial.

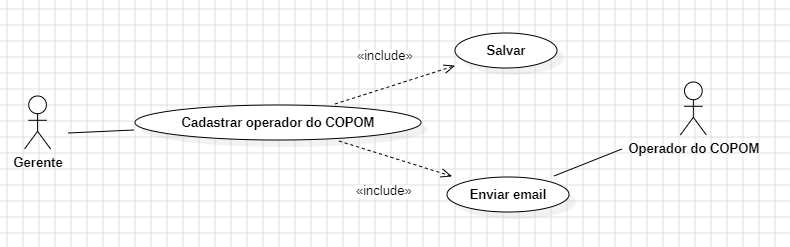
**Pré-condições:** O ator deve estar autenticado no sistema e ter a devida autorização para cadastrar os operadores do COPOM.

**Pós-condições**: Operador cadastrado; operador com acesso ao sistema; novo operador listado na dashboard de controle.

**Fluxo principal**:

1. O superior das equipes (*admin*) clica no botão “Cadastrar Novo Operador” numa *dashboard* de controle;
2. O sistema exibe o formulário de criação de operadores;
3. O admin preenche os campos obrigatório;
4. O sistema valida os dados em tempo real;
5. O admin clica em “Cadastrar”;
6. O sistema envia um e-mail para o operador cadastrado com seu usuário e senha.

Figura 31 - Caso de uso para cadastros de equipes de policiais e operadores do COPOM



Fonte: Autoria própria, 2023

#### [RF003] Iniciar operações

Este caso de uso descreve como um membro da COPOM pode iniciar seu expediente.

**Ator:** Operador do COPOM.

**Prioridade:** Essencial.

**Pré-condições**: Ter sido cadastrado no sistema; fazer autenticação.

**Pós-condições**: Mapa de operações.

**Fluxo principal**:

1. O operador faz a autenticação no sistema;
2. O sistema apresenta uma dashboard para o operador;
3. O operador clica em “Iniciar Operações”;
4. O sistema atualiza o status do operador para “em patrulhamento”;
5. O sistema atualiza a dashboard de controle dos superiores;
6. O sistema disponibiliza o operador para pegar ocorrências.

#### [RF004] Encerrar operações

Este caso de uso descreve como um membro da COPOM pode encerrar seu expediente.

**Ator:** Operador do COPOM.

**Prioridade:** Essencial.

**Pré-condições**: Ter iniciado as operações.

**Pós-condições**: Dashboard de controle do operador.

**Fluxo principal**:

1. O operador clica em “Encerrar Operações”;
2. O sistema atualiza o status do operador para “Inativo”;
3. O sistema atualiza a dashboard de controle dos superiores;
4. O sistema retorna o operador para a dashboard.

#### [RF005] Iniciar patrulhamento

Este caso de uso descreve como a equipe policial iniciar seu expediente.

**Ator:** Equipe de policiais.

**Prioridade:** Essencial.

**Pré-condições**: Ter sido cadastrado no sistema; fazer autenticação.

**Pós-condições**: Mapa de patrulhamento.

**Fluxo principal**:

1. O policial faz a autenticação no sistema;
2. O sistema apresenta uma dashboard para o agente;
3. O policial clica em “Iniciar Patrulhamento”;
4. O sistema sincroniza o início de patrulhamento em todos os membros da equipe individualmente – ou seja, o estado deles passa a ser “em patrulhamento”;
5. O sistema atualiza a dashboard de controle dos superiores;
6. A equipe fica disponível para prestar apoio ou iniciar uma ocorrência.

#### [RF006] Encerrar patrulhamento

Este caso de uso descreve como a equipe policial encerrar seu expediente.

**Ator:** Equipe de policiais.

**Prioridade:** Essencial.

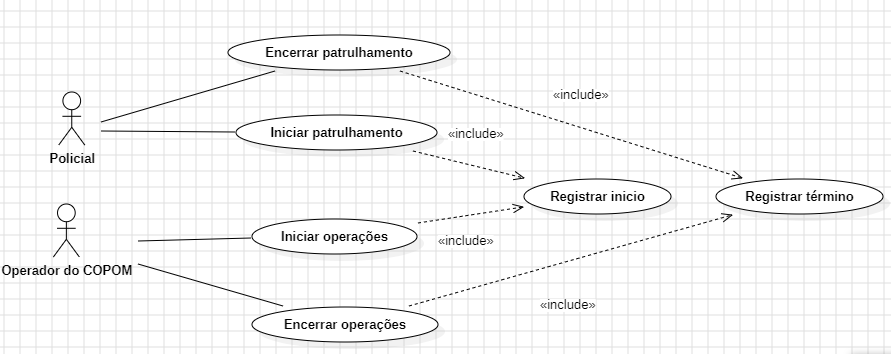
**Pré-condições**: Ter iniciado um patrulhamento.

**Pós-condições**: Dashboard de controle do patrulheiro.

**Fluxo principal**:

1. O policial clica em “Encerrar Patrulhamento”;
2. O sistema sincroniza o término de patrulhamento em todos os membros da equipe individualmente – ou seja, o estado deles passa a ser “inativo”;
3. O sistema atualiza a dashboard de controle dos superiores;
4. A equipe fica indisponível para prestar apoio ou iniciar uma ocorrência e retorna para dashboard.

Figura 32 - Caso de uso para início e término de trabalho dos policiais e operadores do COPOM



Fonte: Autoria própria, 2023

#### [RF007] Iniciar acompanhamento

Este caso de uso descreve como a equipe policial em patrulhamento pode iniciar uma sincronização com o COPOM para evitar a modulação manual.

**Ator:** Equipe de policiais.

**Prioridade:** Essencial.

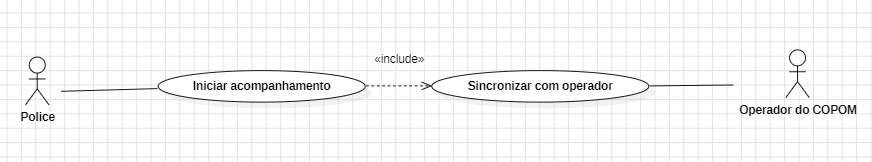
**Pré-condições:** Ter iniciado o patrulhamento (expediente de trabalho).

**Pós-condições**: Dashboard de acompanhamento.

**Fluxo principal**:

1. O policial clica no botão “Iniciar Acompanhamento”;
2. O sistema liga a câmera do policial;
3. O sistema cria um *socket* entre o policial e o operador do COPOM;
4. O sistema abre um mapa e atualiza as coordenadas do policial em tempo real;
5. O sistema envia coordenadas para o COPOM e atualiza o mapa deles em tempo real.

Figura 33 - Caso de uso para início de acompanhamento por parte dos policiais



Fonte: Autoria própria, 2023

#### [RF008] S olicitar apoio

Este caso de uso descreve como a equipe policial em um acompanhamento pode solicitar apoio para o cerco.

**Ator:** Equipe de policiais.

**Prioridade:** Essencial.

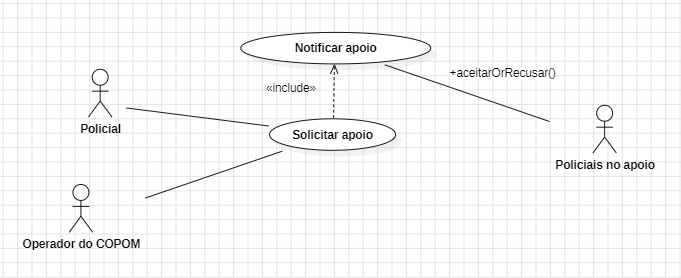
**Pré-condições:** Ter iniciado um acompanhamento (ocorrência).

**Pós-condições**: Dashboard de acompanhamento.

**Fluxo principal**:

1. O policial clica no botão “Solicitar Apoio”;
2. O sistema faz uma varredura de policiais na região que iniciaram o patrulhamento e não estão em nenhuma ocorrência;
3. Os policiais em patrulhamento podem “aceitar” ou “recusar” a solicitação de apoio;
4. O sistema adicionar os policiais que aceitaram o apoio na ocorrência;
5. O sistema sincroniza a posição dos policiais que aceitaram o apoio no mapa.

Figura 34 - Caso de uso para solicitação de apoio por parte dos policiais



Fonte: Autoria própria, 2023

#### [RF009] Acompanhar livestream

Este caso de uso descreve como um operador da COPOM pode abrir, em tempo real, a câmera de um policial que iniciou um acompanhamento (como uma *live stream*). O objetivo é evitar que o policial module manualmente informações sobre as características do veículo e infrator.

**Ator:** Operador do COPOM.

**Prioridade:** Desejável.

**Pré-condições:** Equipe policial ter iniciado um acompanhamento (ocorrência).

**Pós-condições**: Lista com todos os policiais na ocorrência; interface de live stream.

**Fluxo principal**:

1. O operador clica o botão “Assistir” no soldado desejado;
2. Uma interface com o vídeo da câmera do policial é aberta.

### Requisitos não funcionais

#### [NF001] Interface de patrulhamento minimalista

Quando em patrulhamento, os policiais devem manter todo o seu foco no trânsito e em eventuais suspeitos. Quanto menor a quantidade de “clicks” e informações na tela, melhor para o policial.

**Prioridade**: Essencial.

**Casos(s) de uso associado(s)**: [RF004]; [RF005].

#### [NF002] Botão para acionar “iniciar acompanhamento”

Quando um indivíduo desobedecer a ordem de parada e empreender fuga, o policial deve com facilidade ser capaz de clicar em um botão para “iniciar o acompanhamento”. O correto acionamento deve ser respondido com um feedback claro, para que não haja dúvidas de que ele conseguiu iniciar o acompanhamento e uma sincronização com o COPOM está sendo feita. Por fim, erros de clique no botão devem ser minimizados.

**Prioridade**: Importante.

**Casos(s) de uso associado(s)**: [RF005].

#### [NF003] Usar voz para acionar “iniciar acompanhamento”

Quando um indivíduo desobedecer a ordem de parada e empreender fuga, pensando nos policiais que trabalham em motocicletas principalmente, seria interessante que eles não tirassem as mãos do guidão da moto. Para isso, poderia ser possível acionar o “iniciar o acompanhamento” com a voz.

**Prioridade**: Desejável.

**Casos(s) de uso associado(s)**: [RF005].

#### [NF004] Utilização do 5G para comunicação em tempo real

O sistema deve ser capaz de aproveitar a tecnologia de comunicação 5G para fornecer uma conectividade de alta velocidade, baixa latência e confiável entre os policiais envolvidos na perseguição, operadores do COPOM e outros policiais na região. A utilização do 5G garantirá uma transmissão rápida e eficiente das coordenadas e informações gráficas no mapa, permitindo uma colaboração ágil e precisa entre as equipes.

**Prioridade**: Importante.

**Casos(s) de uso associado(s)**: [RF005]; [RF006]; [RF007].

#### [NF005] Manter a sessão em regiões sem cobertura de sinal

O sistema deve ser resiliente e, ao adentrar zonas com baixa disponibilidade de sinal, deve ser capaz de continuar armazenando as coordenadas para enviar para os demais assim que possível.

**Prioridade**: Essencial.

**Casos(s) de uso associado(s)**: [RF005]; [RF006].

#### [NF006] Segurança e prevenção de interceptação de mensagens

O sistema deve ser projetado com medidas robustas de segurança para garantir a confidencialidade, integridade e autenticidade das mensagens transmitidas entre os policiais envolvidos na perseguição, operadores do COPOM e outros policiais na região. Deve-se implementar mecanismos adequados de criptografia, autenticação e controle de acesso para prevenir a interceptação ou acesso não autorizado às informações sensíveis transmitidas.

A segurança da comunicação é de extrema importância para garantir que as informações transmitidas durante uma perseguição sejam acessíveis apenas pelas partes autorizadas. A interceptação de mensagens por indivíduos não autorizados pode comprometer a eficácia da operação, colocar em risco a segurança dos envolvidos e prejudicar a integridade das investigações policiais. Portanto, é essencial implementar medidas de segurança adequadas para proteger a confidencialidade e a privacidade dos dados transmitidos.

**Prioridade**: Essencial.

**Casos(s) de uso associado(s)**: [RF005]; [RF006].

#### [NF007] Quantidade de policiais no apoio

Muitos policiais podem aceitar o apoio solicitado. O sistema deve ser capaz de lidar com um aumento no número de usuários, dispositivos e mensagens transmitidas, garantindo um desempenho consistente e sem degradação significativa em situações de alto tráfego.

**Prioridade**: Essencial.

**Casos(s) de uso associado(s)**: [RF006].

#### [NF008] Baixo tempo de resposta

O sistema deve fornecer uma resposta rápida e eficiente, garantindo tempos de resposta baixos e desempenho otimizado, mesmo em condições de carga intensa ou conexões de rede mais lentas. Porque, assim, o tempo de reação dos policiais no apoio e dos operadores do COPOM será dentro da realidade, melhorando a coordenação e estratégia dos policiais na operação.

**Prioridade**: Essencial.

**Casos(s) de uso associado(s)**: [RF006]; [RF007].

### Matriz de rastreabilidade

Uma matriz de rastreabilidade é uma ferramenta que tem como objetivo principal estabelecer e documentar a relação entre diferentes elementos em um projeto ou sistema. Seu objetivo principal é fornecer uma visão clara e rastreável das conexões entre os requisitos, permitindo assim, por exemplo, descobrir quais requisitos serão afetados caso um outro requisito seja mudado.

Com base nos requisitos funcionais, não funcionais e com a interface com o usuário, a elaboração da matriz de rastreabilidade foi realizada com o intuito de concluir se o sistema atende o que foi solicitado, junto com sua breve descrição, prioridade, dependência e sujeito. Veja a tabela 3 a seguir:

Tabela 6 - Matriz de rastreabilidade do projeto

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Código | | Título | | | | Prioridade | | | Dependência | | | Sujeito | |
| RF001 | | Cadastrar equipe de campo | | | | Essencial | | | \* | | | Gerentes | |
| RF002 | | Cadastrar operador do COPOM | | | | Essencial | | | \* | | | Gerentes | |
| RF003 | | Iniciar operações | | | | Essencial | | | RF002 | | | COPOM | |
| RF004 | | Encerrar operações | | | | Essencial | | | RF003 | | | COPOM | |
| RF005 | | Iniciar patrulhamento | | | | Essencial | | | RF001 | | | Policial | |
| RF006 | | Encerrar patrulhamento | | | | Essencial | | | RF005 | | | Policial | |
| RF007 | | Iniciar acompanhamento | | | | Essencial | | | RF005 | | | Policial | |
| RF008 | | Solicitar apoio | | | | Essencial | | | RF003, RF005, RF007 | | | Policial, COPOM | |
| RF009 | | Acompanhar livestream | | | | Desejável | | | RF003, RF005 | | | COPOM | |
|  | | | | | | | | | | | | | |
| NF001 | | Interface de patrulhamento minimalista | | | | Importante | | | RF005 | | | Policial | |
| NF002 | | Botão para “iniciar acompanhamento” | | | | Importante | | | RF005 | | | Policial | |
| NF003 | | Recurso de voz para “iniciar acompanhamento” | | | | Desejável | | | RF005 | | | Policial | |
| NF004 | | Utilizar 5G para comunicação em tempo real | | | | Importante | | | RF005, RF003, RF008, RF009 | | | Policial, COPOM | |
| NF005 | | Manter sessão em regiões sem cobertura de sinal | | | | Essencial | | | RF005 | | | Policial | |
| NF006 | | Segurança e prevenção de interceptação de eventos | | | | Essencial | | | RF005, RF008 | | | Policial, COPOM | |
| NF007 | | Quantidade de policiais no apoio | | | | Essencial | | | RF008 | | | Policial | |
| NF008 | | Baixo tempo de resposta | | | | Essencial | | | RF005, RF008, RF009 | | | Policial, COPOM | |
|  | | | | | | | | | | | | | |
|  | RF001 | RF002 | RF003 | RF004 | | RF005 | RF006 | | RF007 | RF008 | | RF009 |
| RF001 |  |  |  |  | |  |  | |  |  | |  |
| RF002 |  |  |  |  | |  |  | |  |  | |  |
| RF003 |  |  |  |  | |  |  | |  |  | |  |
| RF004 |  |  |  |  | |  |  | |  |  | |  |
| RF005 |  |  |  |  | |  |  | |  |  | |  |
| RF006 |  |  |  |  | |  |  | |  |  | |  |
| RF007 |  |  |  |  | |  |  | |  |  | |  |
| RF008 |  |  |  |  | |  |  | |  |  | |  |
| RF009 |  |  |  |  | |  |  | |  |  | |  |

Fonte: Autoria própria, 2023

## Desenvolvimento dos requisitos

Para cadastrar uma equipe de campo [RF001], primeiro criou-se um servidor NestJS com o seguinte comando:

Comando 1 - Criar um projeto NestJS

npm i -g @nestjs/cli

nest new [PROJECT\_NAME]

Fonte: NestJS Docs, 2023[[1]](#footnote-1)

O CLI do NestJS irá criar uma estrutura padrão com um módulo raiz chamado “AppModule”. A partir dele, foram criados dois módulos para realização do CRUD de agentes e equipes no sistema, com o seguinte comando:

Comando 2 - Gerar um CRUD com NestJS

nest g resource [RESOURCE\_NAME]

Fonte: Autoria própria, 2023

Para cada módulo, três classes serão geradas seguindo os padrões de design do NestJS (para representa-las, será utilizado o nome de recurso “user”:

* “UserModule”: o módulo raiz do recurso;
* “UserController”: uma *controller* responsável por gerenciar as requisições e respostas HTTP;
* “UserService”: uma camada subsequente da controller, responsável por encapsular as regras de negócio e separa-las do tráfego de entrada.

Código 6 - CRUD gerado com NestJS

import [...];

@Module({

controllers: [UsersController],

providers: [UsersService],

exports: [UsersService],

})

export class UsersModule {}

@Controller('users')

export class UsersController {

constructor(private readonly usersService: UsersService) {}

@Post()

create(@Body() createUserDto: CreateUserDto) {}

@Get()

findAll() {}

@Get(':id')

findOne(@Param('id') id: string) {}

@Patch(':id')

update(@Param('id') id: string, @Body() updateUserDto: UpdateUserDto) {}

@Delete(':id')

remove(@Param('id') id: string) {}

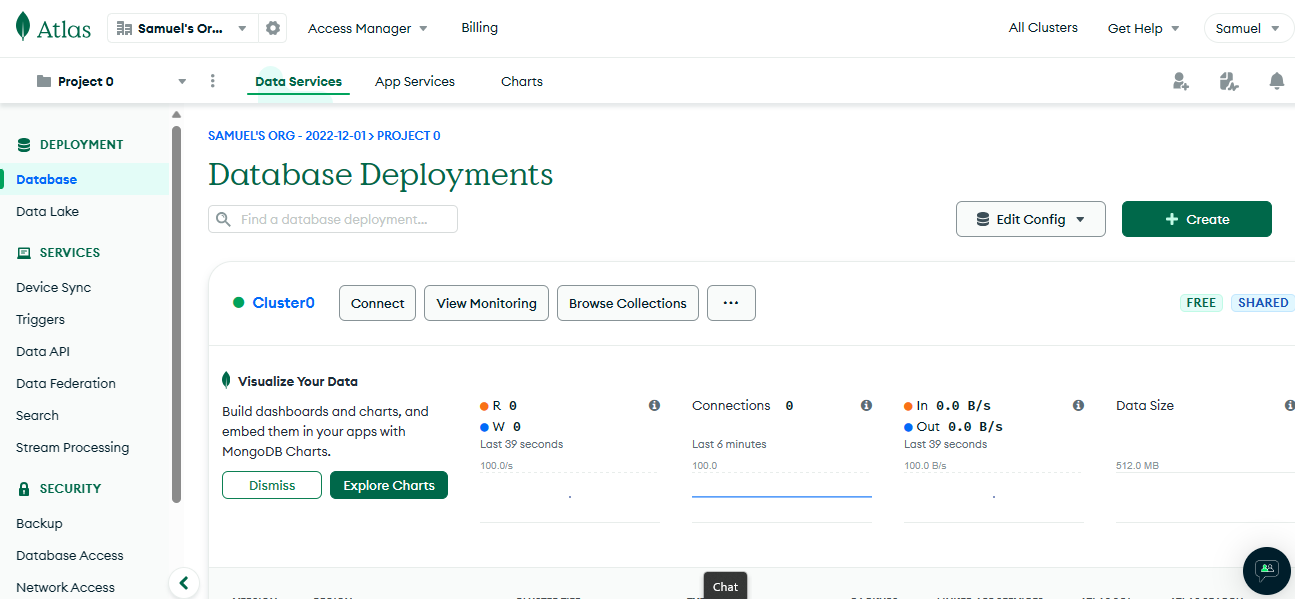
}

Fonte: Autoria própria, 2023

Com as interfaces de CRUD de agentes e equipes criadas, foi necessário criar funcionalidades específicas para gestão de grupos: inserir e remover agentes nos grupos.

Para armazenamento de dados, foi criado um banco MongoDB utilizando o serviço grátis da Atlassian na nuvem:

Figura 35 - Interface de gestão do MongoDB na Cloud da Atlas



Fonte: Autoria própria, 2023

Para conectar o MongoDB com o NestJS, primeiro é necessário instalar as dependências. Neste caso, será utilizado MongoDB com TypeORM:

Comando 3 - Instalar dependências do MongoDB no NestJS

npm i mongodb

npm i typeorm

npm i @nestjs/typeorm

npm i @nestjs/config

Fonte: Autoria própria, 2023

Depois de instaladas, é necessário fazer a configuração na raiz principal da aplicação, o “AppModule”:

Código 7 - Importar o TypeORM no NestJS e fazer a configuração do MongoDB

import [...];

@Module({

imports: [

ConfigModule.forRoot({

isGlobal: true,

}),

TypeOrmModule.forRootAsync({

inject: [ConfigService],

useFactory: (config: ConfigService) => ({

type: 'mongodb',

database: config.get<string>('DB\_NAME'),

url: config.get<string>('DB\_URL'),

synchronize: true,

entities: [User],

}),

}),

UsersModule,

],

controllers: [AppController],

providers: [AppService],

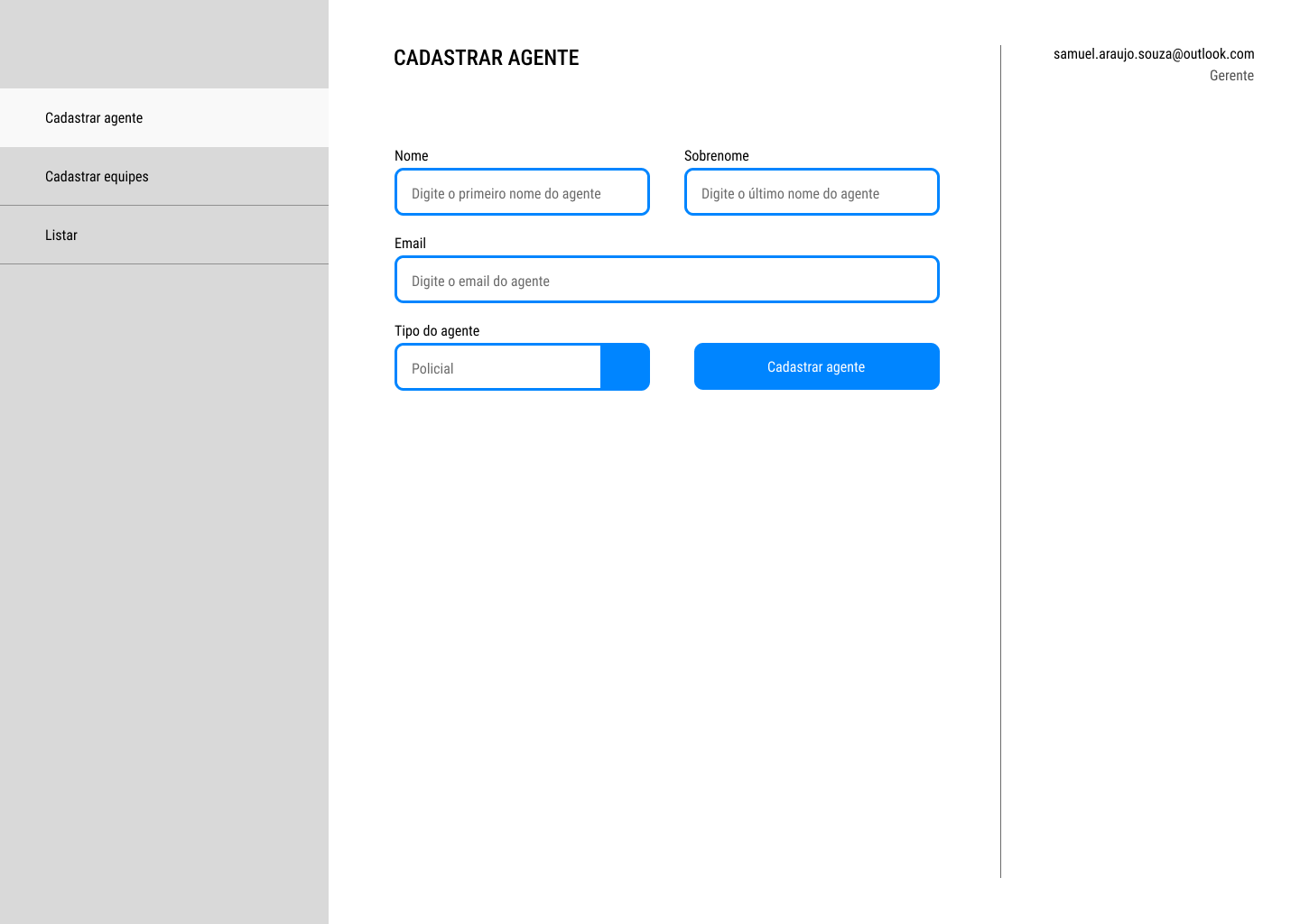
})

export class AppModule {}

Fonte: Autoria própria, 2023

Com o servidor configurado, agora vem a criação da interface de usuário, feita com React DOM e estilizada com CSS puro:

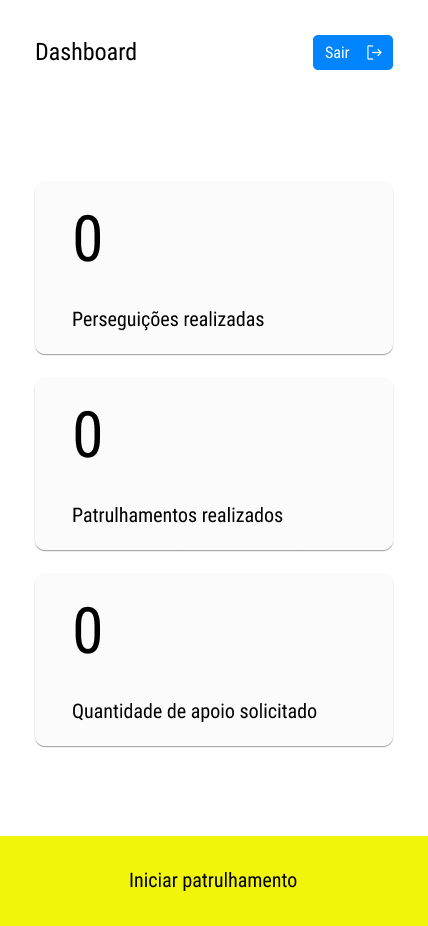
Figura 36 - Interface para cadastro de agentes no React DOM



Fonte: Autoria própria, 2023

Com os agentes cadastrados, eles irão receber uma email de confirmação com as credenciais de acesso padrão. No primeiro login, será solicitado a definição de uma nova senha. Ao fazer login, um policial verá a seguinte tela:

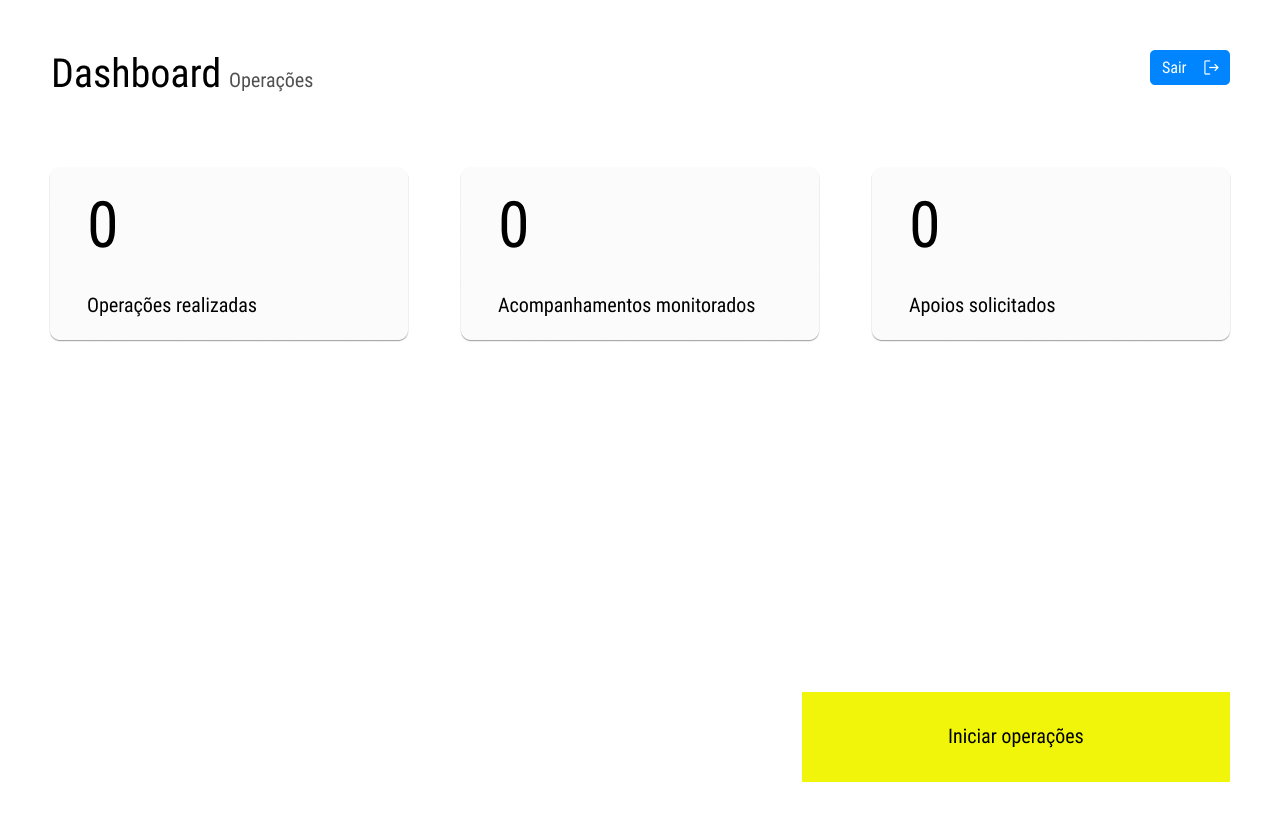
Figura 37 - Dashboard dos policiais no React Native



Fonte: Autoria própria, 2023

Os operadores do COPOM:

Figura 38 - Dashboard dos operadores do COPOM no React DOM



Fonte: Autoria própria, 2023

Ao iniciar um patrulhamento ou operação, ambos irão se conectar com um servidor *websocket* para a que as ações em tempo real aconteçam. Para se conectar, antes é necessário instalar o *client* do websocket em ambos (React Native e DOM):

Comando 4 – Instalar o socket.io cliente no React Native e DOM

npm i socket.io-client

Fonte: NPM, 2023[[2]](#footnote-2)

Para que o se conecte e crie uma sessão com o servidor socket, é necessário fazer a seguinte configuração:

Código 8 - Instanciar conexão com servidor websocket com o hook "useMemo"

const socket = useMemo(() => {

return io(process.env.EXPO\_PUBLIC\_SOCKET\_SERVER as string, {

query: { clientInitialRoom: [CLIENT\_TYPE] }

}).connect()

}, []);

Fonte: Autoria própria, 2023

O *hook* “useMemo” do React garante que apenas uma instância de conexão com o servidor será criada, evitando o desperdício de memória. Cada cliente, precisa passar o seu tipo no parâmetro “query”. Se um operador do COPOM, então “operations”, se não, “patrolling”. Isso definirá em qual sala o cliente será colocado.

Para criar um servidor websocket, primeiro deve-se instalar as dependências com o seguinte comando:

Comando 5 - Instalar socket.io no NestJS

npm i --save @nestjs/websockets @nestjs/platform-socket.io

Fonte: NestJS Docs, 2023[[3]](#footnote-3)

Para gerar um módulo para o socket no servidor, o seguinte comando é necessário:

Comando 6 - Gerar um gateway socket no NestJS

nest g gateway [RESOURCE\_NAME]

Fonte: Autoria própria, 2023

A seguinte classe é gerada:

Código 9 - Classe Websocket Gateway no NestJS

import [...]

@WebSocketGateway(0, { cors: { origin: '\*', }, })

export class UsersGateway {

@WebSocketServer()

private server: Server;

}

Fonte: Autoria própria, 2023

Para ser capaz de fazer a troca de mensagens entre os operadores do COPOM e os policiais em campo, os seguintes eventos foram criados com o “SubscribeMessage”:

Código 10 - Eventos para gerenciar a troca de mensagens entre o COPOM e os policias em campo

@WebSocketGateway(0, { cors: { origin: '\*', }, })

export class UsersGateway {

@WebSocketServer()

private server: Server;

@SubscribeMessage('patrollingPosition')

handlePatrollingPosition(

@ConnectedSocket() client: Socket,

@MessageBody() position: number[],

) {}

@SubscribeMessage('finishedPatrolling')

handleFinishedPatrolling(@ConnectedSocket() client: Socket) {}

@SubscribeMessage('startedChasing')

async handleStartedChasing(@ConnectedSocket() client: Socket) {}

@SubscribeMessage('acceptOperations')

handleAcceptOperations(

@ConnectedSocket() client: Socket,

@MessageBody() chasingId: string,

) {}

}

@SubscribeMessage('chasingPosition')

handleChasingPosition(

@ConnectedSocket() client: Socket,

@MessageBody() position: number[],

) {}

@SubscribeMessage('finishedChasing')

handleFinishedChasing(@ConnectedSocket() client: Socket) {}

@SubscribeMessage('supportRequest')

handleSupportRequest(@ConnectedSocket() client: Socket) {}

@SubscribeMessage('acceptSupport')

handleAcceptSupport(

@ConnectedSocket() client: Socket,

@MessageBody() chasingId: string,

) {}

@SubscribeMessage('supportPosition')

handleSupportPosition(

@ConnectedSocket() client: Socket,

@MessageBody() { position, chasingId }

) {}

@SubscribeMessage('getIntoRoom')

handleGetIntoRoom(

@ConnectedSocket() client: Socket,

@MessageBody() room: string,

) {}

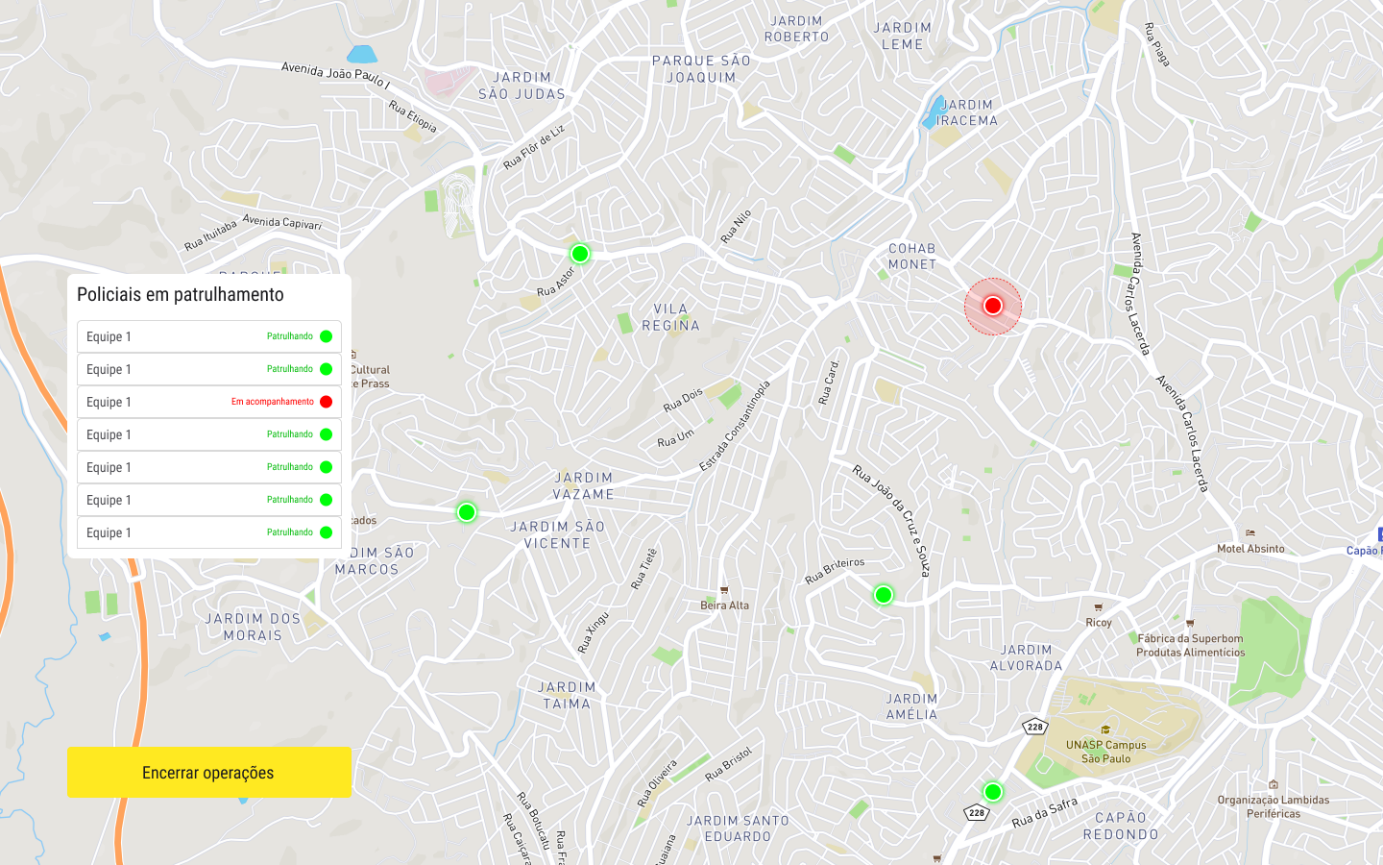
Fonte: Autoria própria, 2023

Detalhes de implementação foram omitidos para uma melhor legibilidade, deixando apenas a assinatura do método. Aqui segue uma lista explicando o que cada evento faz:

* “'patrollingPosition”: responsável por transmitir a posição de todos os policias que iniciaram um patrulhamento, na sala “patrulhamento”, para os operadores do COPOM renderizarem no cliente;
* “'finishedPatrolling”: é responsável por avisar os operadores do COPOM que um policial terminou o patrulhamento, assim, o policial pode ser retirado do mapa;
* “startedChasing”: é responsável por tirar um policial da sala de “patrulhamento” e colocar em “acompanhamento.[ID\_PERSONALIZADO]” assim que ele iniciar um acompanhamento. Obrigatoriamente, esse método conecta o policial com um operador do COPOM;
* “'acceptOperations”: é responsável pelo aceite do operador do COPOM para um acompanhamento em específico, retirando ele da sala “operações” e colocando-o em “acompanhamento.[ID\_PERSONALIZADO]”;
* “chasingPosition”: é responsável por fazer o envio de coordenadas do policial que está perseguindo para que todos possam ver sua posição;
* “finishedChasing”: é responsável por encerrar um acompanhamento e retornar todo mundo para suas respectivas salas;
* “'supportRequest”: é responsável por solicitar apoio por parte de um policial num acompanhamento, enviando a notificação para todos os policiais em patrulhamento;
* “acceptSupport”: é responsável por lidar com o aceite de um policial patrulhando para entrar num acompanhamento como apoio;
* “supportPosition”: é responsável por transmitir a posição de todos os policias que estão no apoio em uma perseguição;
* “getIntoRoom”: é responsável por inserir uma agente em uma determinada sala.

A seguir, a tela de um operador do COPOM que iniciou suas operações:

Figura 39 - Mapa de operações do COPOM

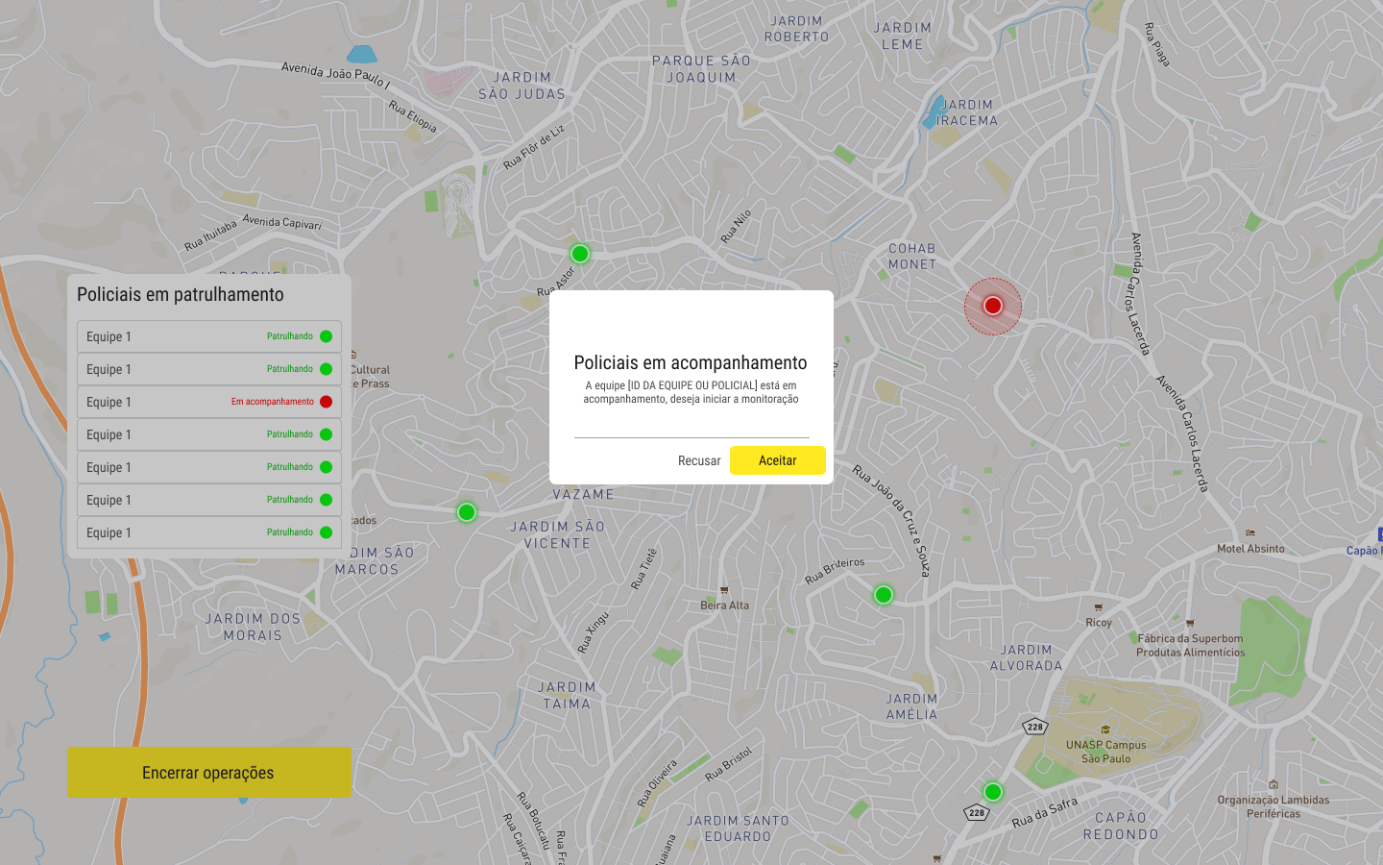


Fonte: Autoria própria, 2023

Na esquerda, um pequeno painel que lista todos os policias em patrulhamento de forma paginada. Ao clicar em uma equipe, a o mapa se centraliza neles e fixa a visão. Cada policial tem um status, podendo este ser “patrulhando” ou “em acompanhamento”; as cores mudam de maneira correspondente.

Se um policial inicia um acompanhamento, o operador recebe a notificação:

Figura 40 - Notificação de início de acompanhamento para o COPOM

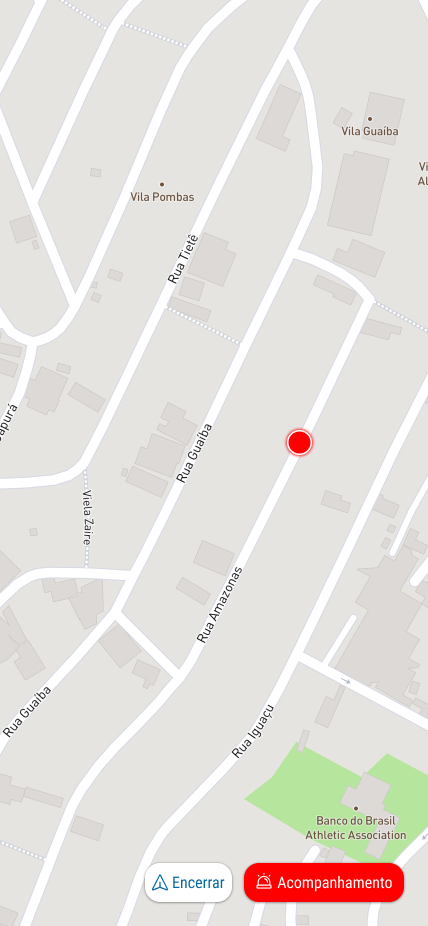


Fonte: Autoria própria, 2023

Ao receber essa notificação, o operador tem a possibilidade de aceitar ou recusar.

Já um policial quando inicia um patrulhamento visualiza a seguinte tela:

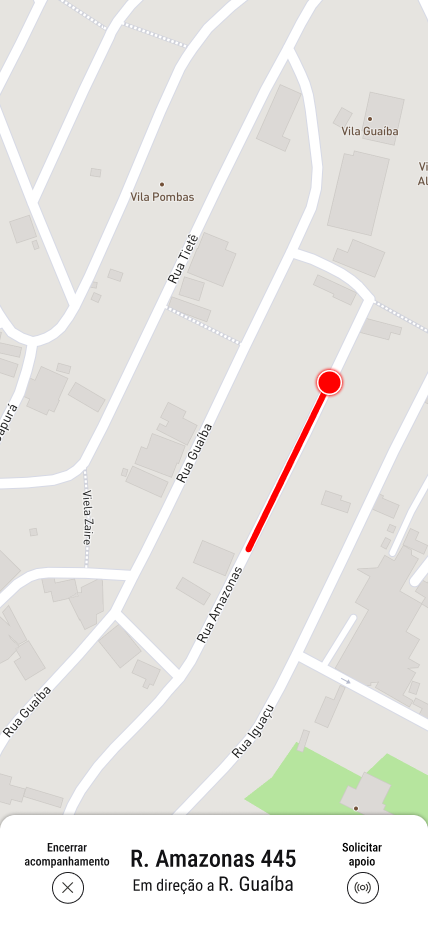
Figura 41 - Mapa de patrulhamento do policial



Fonte: Autoria própria, 2023

A qualquer momento o policial pode encerrar seu patrulhamento ou iniciar uma perseguição. Ao iniciar um acompanhamento, a seguinte tela é renderizada:

Figura 42 - Estado de acompanhamento do policial



Fonte: Autoria própria, 2023

Nesse estado, o policial pode encerrar seu acompanhamento ou ainda solicitar apoio. Ao solicitar apoio, os policiais em patrulhamento receberão uma notificação, podendo recusá-la ou aceitar. Na imagem abaixo é possível ver essa notificação:

Figura 43 - Notificação de solicitação de apoio quando em patrulhamento



Fonte: Autoria própria, 2023

1. Disponível em: https://docs.nestjs.com/first-steps. Acesso em: 30 de setembro de 2023. [↑](#footnote-ref-1)
2. Disponível em: https://www.npmjs.com/package/socket.io-client. Acesso em: 30 de setembro de 2023 [↑](#footnote-ref-2)
3. Disponível em: https://docs.nestjs.com/websockets/gateways. Acesso em: 30 de setembro de 2023 [↑](#footnote-ref-3)