



Projeto final

Sistema de gerenciamento de grupos
rebeldes e ameaças.

003-1040559

1250 003-77156.8

1760 0009-14563.7

73273



CONTEXTO

- República dissolvida e perseguição de resistentes.
- Incapacidade de comunicação entre regiões pelo perigo de locomoção.
- Necessidade de uma forma de comunicação à distância e segura entre os núcleos rebeldes.

OBJETIVO

- Desenvolvimento de um software que facilite a comunicação entre grupos rebeldes.





01

02

03

04

05

06



O SOFTWARE

01

02

03

04

05

06

- Baseado em um **mapa interativo** para cadastro e visualização simplificada de ameaças e grupos rebeldes.
- Possibilita identificação de locais com perigo de se transitar e locais de apoio, de núcleos rebeldes.
- Assim, **facilita** a locomoção **segura** e o **acolhimento** de refugiados.





01

02

03

04

05

06



FUNCIONALIDADES PRETENDIDAS

01

02

03

04

05

06

1. Mapa interativo.

- a. Visualização geral de ameaças e rebeldes cadastrados.
- b. Detalhamento curto dos eventos.

2. Cadastro de eventos.

- a. Rebeldes.
- b. Ameaças.
 - i. Patrulha.
 - ii. Blitz.
 - iii. Recebimento de carga.
 - iv. Combate.
 - v. Marcha.

3. Listagem de cadastrados.

- a. Possibilidade de maior detalhamento.
- b. Excluir evento.





01

02

03

04

05

06



ORGANIZAÇÃO DO PROJETO

01

02

03

04

05

06

1. Interface Gráfica: telas, uso do framework Swing.
 - a. Tela de Login.
 - b. Tela inicial.
 - c. Tela do mapa.
 - d. Tela de cadastro.
 - e. Tela de listagem.
2. Componentes: entidades do projeto.
 - a. Interface Mapeável e classe abstrata Cadastrável.
 - b. Classes.
 - i. Ameaças.
 - ii. Rebeldes.
3. Services: classes de apoio.
 - a. Classe Útil.



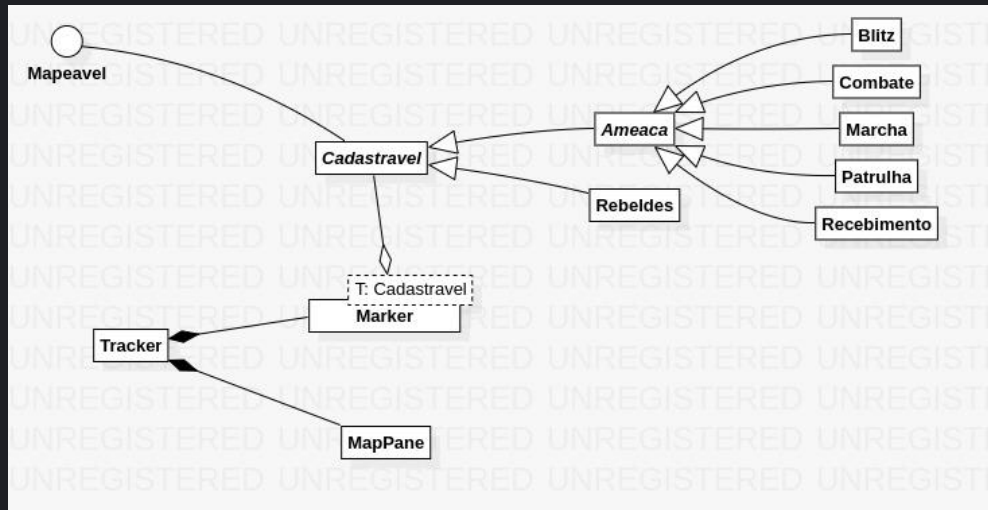


Diagrama de classes UML

Diagrama simplificado que representa a organização do projeto.

LOGIN

- Primeira tela.
- Segurança
 - Acesso do sistema apenas com senha.
 - Encriptação.

```
Login.java

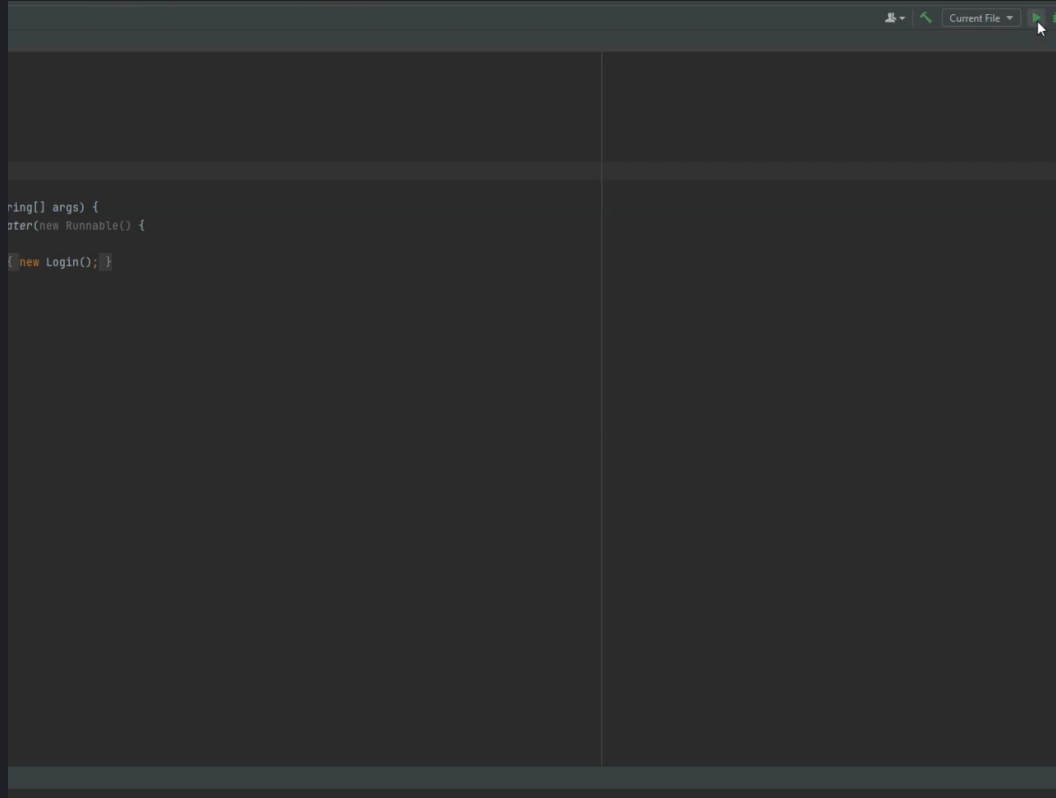
button.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String senhaDigitada = campoSenha.getText();
        String senhaCriptografada = criptografar(senhaDigitada);

        if (senhaCriptografada.equals(passwd)) {
            Tracker tracker = new Tracker();
            tracker = Util.readFile();

            new TelaInicial(tracker);
            dispose();
        }
        else
        {
            JOptionPane.showMessageDialog(new JFrame(), "SENHA INCORRETA", "ERRO", JOptionPane.ERROR_MESSAGE);
            System.exit(0);
        }
    }
});

public static String criptografar(String senha) {
    Base64.Encoder encoder = Base64.getEncoder();
    return encoder.encodeToString(senha.getBytes());
}
```

LOGIN



The image shows a screenshot of an IDE window with a dark theme. The window has a title bar with a user icon, a back arrow, and a dropdown menu labeled "Current File". The main area displays Java code for a `Login` class. The code is as follows:

```
ring[] args) {  
    ster(new Runnable() {  
        { new Login(); }  
    }  
}
```

003-1040559

1250 003-77156.8

1760 0009-14563.7

73273

TELA INICIAL

VISUALIZAR MAPA
CADASTRAR
VISUALIZAR LISTA
SAIR

003-1040559

1250 003-77156.8

1760 0009-14563.7

73273

TELA INICIAL – IMPLEMENTAÇÃO

```
TelaInicial.java

public TelaInicial(Tracker tracker){
    JButton botaoMapa = new JButton();

    try {
        Image botaoMapaImg = ImageIO.read(getClass().getResource("/imagens/visualizar mapa.png"));

        botaoMapa.setIcon(new ImageIcon(botaoMapaImg));
        botaoMapa.setBackground(Color.black);
        botaoMapa.setBorderPainted(false);
        botaoMapa.setFocusPainted(false);
        botaoMapa.setOpaque(false);
        botaoMapa.setContentAreaFilled(false);

        Image botaoMapaImgHover = ImageIO.read(getClass().getResource("/imagens/visualizar mapa hover.png"));
        botaoMapa.setRolloverIcon(new ImageIcon(botaoMapaImgHover));
        botaoMapa.setPressedIcon(new ImageIcon(botaoMapaImgHover));
    } catch (IOException e) {
        throw new RuntimeException(e);
    }

    botaoMapa.addActionListener(e → {
        tracker.run();
    });
}
```



MAPA

- Tela do Mapa.
- Apresenta cadastrados.
- Breve explicação.
- Fundo + marcadores + gerenciamento.



MAPA

MAPPANE

- Classe que representa o mapa em si (imagem).
- Estende JLayeredPane.

MARKERS

- Classe que representa marcadores do mapa.
- Classe genérica <T extends Cadastravel>
- Estende JLabel.

TRACKER

- Classe que gerencia o mapa.
 - Cadastro e Exclusão.
- Contém referência à MapPane a à lista de Markers cadastrados.
- Engloba em si o armazenamento dentro de classes do projeto.

```
Tracker.java

public void run() {
    paneMapa = new MapPane();

    frame = new JFrame("Mapa");
    frame.setUndecorated(true);
    frame.setSize(1200, 900);
    frame.setShape(new Rectangle2D.Double(10, 10, paneMapa.getWidth(), paneMapa.getHeight(), 50, 50));
    frame.setLayout(new BorderLayout());
    frame.setResizable(false);
    frame.setLocationRelativeTo(null);
    frame.setVisible(true);

    frame.add(paneMapa, BorderLayout.CENTER);

    JButton voltar = new JButton();
    try {
        BufferedImage img = ImageIO.read(getClass().getResource("/imagens/botao-voltar-amarelo.png"));
        voltar.setIcon(new ImageIcon(img));
        voltar.setBorderPainted(false);
        voltar.setFocusPainted(false);
        voltar.setOpaque(false);
        voltar.setContentAreaFilled(false);
        voltar.setBorder(BorderFactory.createEmptyBorder(0, 0, 0, 0));
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
    voltar.addActionListener(e -> {
        paneMapa = null;
        frame.dispose();
    });

    paneMapa.setLayout(null);
    paneMapa.add(voltar);
    voltar.setBounds(new java.awt.Rectangle(20, 20, 50, 50));

    refresh();
}

public void refresh() {
    for (Marker atual : markers) {
        atual.setToolTipText("<html>" + atual.getObjeto().getNomeClasse() + "<br>" +
            atual.getObjeto().getDescricaoCurta() + "</html>");
        UIManager.put("ToolTip.background", new Color(249, 224, 124));
        UIManager.put("ToolTip.foreground", Color.BLACK);
        UIManager.put("ToolTip.font", new Font("Arial", Font.PLAIN, 12));

        atual.setSize(atual.getPreferredSize());
        atual.setLocation(atual.getObjeto().getCoordenadaX(), atual.getObjeto().getCoordenadaY());

        paneMapa.add(atual);
    }
}
```



CADASTRAR

- Inserção de informações básicas.
- Seleção do tipo para informações específicas.
- Submissão e repasse dos dados para armazenamento.



CADASTRAR - IMPLEMENTAÇÃO

```
Cadastrar.java

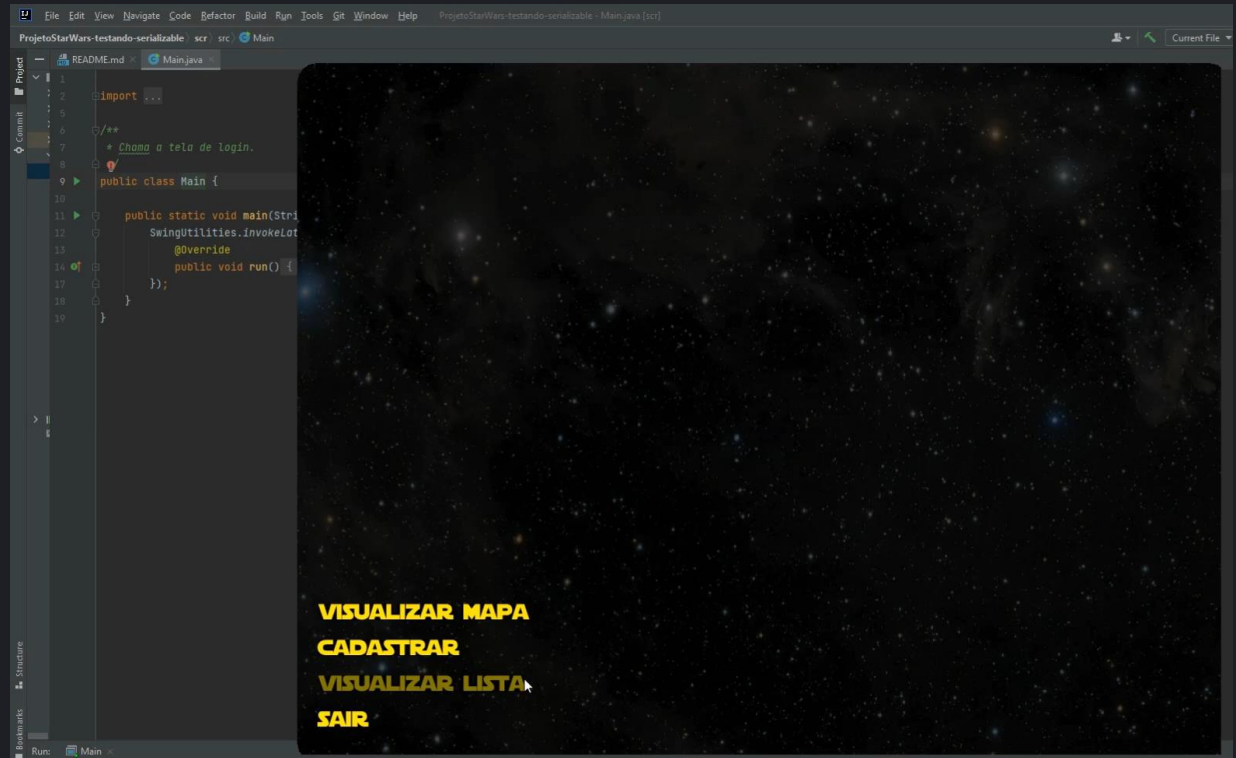
public Cadastrar(Tracker tracker) throws HeadlessException {
    ....
    itensBox.addItemListener(new ItemListener() {
        @Override
        public void itemStateChanged(ItemEvent e) {
            if(e.getStateChange() == ItemEvent.SELECTED) {
                String selecionado = (String) itensBox.getSelectedItem();
                cardPanel.add(form(selecionado, tracker), selecionado);
                cardlayout.show(cardPanel, selecionado);
            }
        }
    });
    ....
}

private JPanel form(String selecionado, Tracker tracker) {
    if (selecionado.equalsIgnoreCase("Rebelde")) {...}
    else if (selecionado.equalsIgnoreCase("patrulha")) {...}
    else if (selecionado.equalsIgnoreCase("Combate")) {...}
    else if (selecionado.equalsIgnoreCase("Marcha")) {...}
    else if (selecionado.equalsIgnoreCase("Recebimento")) {...}
    else {...}

    returnedPanel.setOpaque(false);
    return returnedPanel;
}
```

LISTAGEM DE CADASTRADOS

- Listagem de todos os cadastrados.
- Informações básicas.
- Informações detalhadas.
- Excluir cadastro.



LISTAGEM - IMPLEMENTAÇÃO

```
ListarInformacoes.java

private void percorreCadastrados(Tracker tracker){
    for(Marker atual : tracker.getMarkers()){
        Cadastravel objeto = atual.getObjeto();

        JPanel principal = new JPanelArredondado(this.getForeground(), objeto, tracker);

        JPanel infosBasicas = new JPanel();

        JLabel nome = new JLabel("Nome: " + objeto.getNomeClasse());
        JLabel imagem = new JLabel();
        try {
            ImageIcon img = new ImageIcon(ImageIO.read(getClass().getResource(objeto.getCaminhoImagem())));
            imagem.setIcon(img);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
        JLabel coordenadas = new JLabel("Coordenadas: ( " + objeto.getCoordenadaX() + " , " + objeto.getCoordenadaY() + " )");
        JLabel numero = new JLabel("Número: " + objeto.getNumeros());
        JTextArea descricaoLonga = new JTextArea( "Descrição: " + objeto.getDescricaoLonga(), 5, 25);
        descricaoLonga.setLineWrap(true);

        infosBasicas.add(Box.createRigidArea(new Dimension(25, 10)));
        infosBasicas.add(nome);
        infosBasicas.add(coordenadas);
        infosBasicas.add(numero);

        principal.add(infosBasicas, BorderLayout.WEST);
        principal.add(imagem, BorderLayout.EAST);
        principal.add(descricaoLonga, BorderLayout.SOUTH);

        fundo.add(principal);
    }
}

private class JPanelArredondado extends JPanel{

    public JPanelArredondado(Color corOriginal, Cadastravel objeto, Tracker tracker){
        this.addMouseListener(new JPanelMouseListener(corOriginal, tracker));
        this.objeto = objeto;
    }

    @Override
    protected void paintComponent(Graphics g) {...}
}
```

LISTAGEM - IMPLEMENTAÇÃO

```
ListarInformacoes.java

private class PanelMouseListener extends MouseAdapter {
    @Override
    public void mouseClicked(MouseEvent e) {
        PanelArredondado panel = (PanelArredondado) e.getComponent();
        Cadastravel objeto = panel.objeto;
        criaFrame(objeto);
    }

    @Override
    public void mouseEntered(MouseEvent e) {...}
    public void mouseExited(MouseEvent e) {...}

    private void criaFrame(Cadastravel objeto){
        JFrame frame = new JFrame(objeto.getNomeClasse());
        frame.setSize(new Dimension(350, 270));

        JPanel infosBasicas = new JPanel();
        JPanel infosEspecificas = new JPanel();

        if(objeto instanceof Ameaca){
            JLabel gravidade = new JLabel("Gravidade da ameaça: " + ((Ameaca) objeto).getGravidade());
            infosEspecificas.add(gravidade);
            if(objeto instanceof Blitz){
                JLabel veiculo = new JLabel("Veículo: " + ((Blitz) objeto).getVeiculo());
                infosEspecificas.add(veiculo);
            }
        }

        JButton botaoExcluir = new JButton();
        botaoExcluir.addActionListener(e → {
            tracker.excluir(objeto);
            frame.dispose();
        });

        frame.add(infosBasicas, BorderLayout.NORTH);
        frame.add(infosEspecificas, BorderLayout.CENTER);
        frame.add(botaoExcluir, BorderLayout.EAST);
        frame.setVisible(true);
    }
}
```



SAIR

- Fecha tela e para o programa.
- Dados são **salvos** em arquivos.



ARMAZENAMENTO

Leitura de Arquivo

Leitura de arquivo de Object Stream ao logar no sistema.
Classe `Tracker`.

Escrita no Arquivo

Escrita da classe `Tracker` no arquivo serializável ao sair do sistema.

Senha

Armazenada `encriptografada` em arquivo de texto.
Lida no início do programa.

LIMITAÇÕES E PLANOS FUTUROS

- Atualmente, o software só funciona no planeta **Coruscant**.
- **Inclusão** de novos planetas para cadastro de eventos, permitindo ao usuário a escolha de qual planeta visualizar.
- Acolhimento de grupos rebeldes **espalhados** pela galáxia.
- Com a expansão, aprimorar a maneira de visualização dos dados.