Utilizando o Sentry para Monitoração de Falhas de Execução no Aplicativo EducAPI Manager: Um Relato de Experiência

Eduardo Freire da Costa[1]

Departamento de Ciências Exatas (DCX) –
Universidade Federal da Paraíba (UFPB) - Campus IV
Cep 58297-000 – Rio Tinto – PB – Brasil

eduardo.freire@dcx.ufpb.br

Abstract. *Several issues can occur after software delivery. Considering that, it is important to monitor the occurrence of these issues quickly to support software maintenance and correct possible software defects. Many companies use fault monitoring tools in order to monitor serious failures, such as the Sentry tool. This technical report aims to use the Sentry tool in the EducAPI-Manager application, capture and report possible failures and present the experience obtained in order to support other developers who want to use this tool in Android applications.*

Resumo. *Diversos problemas podem ocorrer após a entrega do software. Considerando isso, é importante monitorar a ocorrência desses problemas de forma rápida para dar suporte à manutenção de software e corrigir possíveis defeitos. Muitas empresas utilizam ferramentas de monitoração de falhas, a fim de monitorar falhas graves. Uma dessas ferramentas, por exemplo, é o Sentry. Este relatório técnico tem como objetivo utilizar a ferramenta Sentry no aplicativo EducAPI-Manager, capturar possíveis falhas, relatar e registrar a experiência obtida de forma que possa apoiar outros desenvolvedores que desejam utilizar esta ferramenta em aplicativos Android.*

1. Introdução

É importante evitar que qualquer software, depois de entregue aos usuários, apresente falhas de execução ou chegue a parar de funcionar, o que é comumente chamado de "*crash*[2]" ou "*fatal exception*". Após a entrega do software, diversos problemas podem ocorrer, isto é, falhas resultantes de erros durante a etapa de desenvolvimento ou mesmo falhas que podem acontecer em apenas alguns dispositivos e não previstas anteriormente, por exemplo. Para isso, é importante monitorar a ocorrência desses eventos para ajudar na manutenção e aprimoramento de sistemas ou aplicativos, pois é necessário ter o *feedback* o mais rápido possível com relação a possíveis problemas na execução.

---

[1] Trabalho de conclusão de curso, sob orientação da professora Ayla Débora Dantas de Souza Rebouças submetido ao Curso de Licenciatura em Ciência da Computação do Centro de Ciências Aplicadas e Educação (CCAE) da Universidade Federal da Paraíba, como parte dos requisitos necessários para obtenção do grau de LICENCIADO EM CIÊNCIA DA COMPUTAÇÃO.

[2] Developer Android: Disponível em: https://developer.android.com/topic/performance/vitals/crash/. Acesso em 22 Setembro de 2022

A manutenção, segundo Pressman (2021), é um processo que continua mesmo após o software ser entregue ao cliente, fazendo assim parte do ciclo de vida do software. A construção inicial de um software pode durar meses ou anos, e a manutenção e modificação do mesmo pode chegar a uma década ou mais, ou seja, é uma fase que exige grande parte do esforço e tempo da empresa. Alguns engenheiros de software acreditam que a maior parte do dinheiro gasto em um artefato será nas atividades de manutenção [Pressman et al. 2021]. Segundo Morais (2020), não é raro uma empresa de software despender de 60% a 70% de todos os recursos com manutenção de software.

Para apoiar o processo de manutenção, grandes e pequenas empresas de desenvolvimento de software podem utilizar ferramentas de monitoramento de *crashes* para que os times de desenvolvimento sejam avisados e possam rastrear eventos anormais. Algumas dessas ferramentas são o *Crashlytics*[3] e o *Sentry*[4]. Neste trabalho se focou na ferramenta *Sentry*, que além de monitorar falhas em softwares, também tem as opções de capturar, registrar e organizar essas falhas em painéis (*dashboards*) com informações mais detalhadas.

O objetivo deste trabalho com formato de relatório técnico é utilizar uma ferramenta de análise de falhas de execução (*Sentry*) em um aplicativo *Android* sendo exercitadas suas funcionalidades, capturar possíveis falhas e reportar a experiência obtida de forma a apoiar desenvolvedores no processo de manutenção corretiva de software que precisem utilizar esse tipo de ferramenta. O *EducAPI-Manager* foi o aplicativo *Android* escolhido. Ele consiste na interface para dispositivos móveis *Android* de um sistema colaborativo para apoiar a alfabetização. Com ele é possível cadastrar, consultar, alterar e excluir dados da base colaborativa do serviço *EducAPI* com diferentes imagens e palavras para atividades de alfabetização. O aplicativo foi desenvolvido utilizando *Kotlin*[5] e *Java*[6] como linguagens de programação [Ludgério et al. 2021] como parte de projetos de pesquisa e extensão da UFPB-Campus IV. Atualmente o aplicativo é mantido pelo projeto *Apps4Society*, que tem como objetivo principal estimular o desenvolvimento de aplicativos que possam impactar positivamente a sociedade [*Apps4Society* 2022].

. Ele foi escolhido para este trabalho por ser implementado em linguagens que vêm sendo comumente utilizadas no mercado, além do fato do autor deste trabalho apresentar familiaridade com o código do aplicativo. Pretende-se dessa forma mostrar a importância de ferramentas como o *Sentry* para a manutenção de software considerando aspectos como monitoramento e análise de relatórios de falhas. Além disso, espera-se apresentar neste relatório técnico os passos necessários para utilização da ferramenta na análise das falhas de um aplicativo.

Para atingir os objetivos previstos neste trabalho, foram seguidos os seguintes passos metodológicos: i) levantamento da literatura; ii) preparação do ambiente para utilização do *Sentry* no EducAPI-Manager; iii) identificação e registro dos passos da integração da ferramenta no aplicativo; iv) aplicação da ferramenta no aplicativo; v) registro e análise dos dados coletados; vi) relato da experiência com uso da ferramenta e sua contribuição para a manutenção de software.

---

[3] Crashlytics. Disponível em: https://firebase.google.com/docs/crashlytics/. Acesso em 3 set. 2022
[4] Sentry: Disponível em: https://sentry.io/. Acesso em 3 set. 2022
[5] Kotlin: Disponível em: https://kotlinlang.org/. Acesso em 7 set. 2022
[6] Java: Disponível em: https://www.java.com/pt-BR/. Acesso em 7 set. 2022

As demais seções deste artigo estão organizadas conforme descrito a seguir: a Seção 2 apresenta a fundamentação teórica e ferramentas de monitoração de falhas de execução; a Seção 3 apresenta o sistema EducAPI e o aplicativo EducAPI-Manager; a Seção 4 apresenta a metodologia utilizada no trabalho; a Seção 5 apresenta os resultados; e a Seção 6 apresenta as conclusões e propostas de trabalhos futuros.

2. Manutenção de software e ferramentas de monitoração de falhas de execução

Nesta seção serão apresentados conceitos relacionados à manutenção de software e ferramentas utilizadas para monitoração de falhas de execução de software.

2.1 Manutenção de software

É muito difícil existir software perfeito, já que mesmo que o desenvolvedor siga à risca algumas regras de qualidade de software e padrões de projeto, mudanças ou situações não previstas podem sempre ocorrer e gerar falhas. É por isso que se faz necessária a manutenção de software. A manutenção é definida como o conjunto de atividades necessárias para manter o software operacional após ele ser aceito e entregue (lançado) no ambiente do usuário [Pressman et al. 2021]. Um dos tipos de manutenção de software é a manutenção corretiva, que é o tipo de manutenção no qual este trabalho se foca. A manutenção corretiva, segundo Pressman (2021) é a modificação reativa do software para consertar problemas descobertos após o software ter sido entregue ao consumidor final.

Todo software passa por um ciclo de vida. Ele nasce, tem o seu período de existência e algum dia pode morrer. Um software pode chegar a morrer por diversos fatores, seja por ele não ser mais necessário e vir a ser descontinuado por seus desenvolvedores ou por falta de manutenção, por exemplo.

Após o software ser entregue ao cliente, ele passa pela etapa de manutenção e melhoria. Nessa etapa é necessário ter o *feedback* o mais rápido possível com relação a possíveis problemas na execução. Quando o software já foi entregue ao cliente, é normal que apresente falhas (também conhecidas como defeitos ou *bugs*) [Rebouças 2010]. Por exemplo, o usuário está utilizando um aplicativo e de repente este aplicativo para de funcionar por algum problema. É importante que o desenvolvedor saiba sobre esses problemas o quanto antes.

Para isso, o cliente pode simplesmente entrar em contato com a equipe de suporte e tentar informar o fato, que poderá ser encaminhado para os desenvolvedores. Porém, alguns problemas podem ocorrer apenas com alguns usuários ou dispositivos. Tomemos como exemplo aplicativos para a plataforma *Android*, que é o sistema operacional mais utilizado em dispositivos móveis [Statista 2022]. O mesmo software pode ser instalado em dispositivos de diferentes marcas, processadores, quantidade de memória e apresentar falhas só em alguns casos.

Outra questão importante é que alguns usuários não entram em contato com a equipe de suporte, ou seja, aguardam o aplicativo voltar ao normal, desinstalam ou partem para outro aplicativo semelhante.

O tempo até o desenvolvedor encontrar o problema pode ser rápido ou durar muito tempo. Isso ocorre pois nem sempre é fácil replicar o problema, corrigir e testar. Se esse processo de manutenção tiver o apoio de ferramentas, uma empresa pode reduzir seus custos e problemas podem ser resolvidos mais rapidamente.

Considerando que há muitos fatores problemáticos que podem ocorrer no ciclo de vida do software, se faz necessária a manutenção corretiva, pois ela vai garantir de forma reativa a manutenção do software, apoiando a correção de *bugs* que não foram vistos antes da entrega e possíveis novos *bugs* que surgiram após a entrega, independentemente da complexidade do *bug* ou falhas que podem ocorrer após adaptações do código do sistema.

2.2 Ferramentas de Monitoração e Visualização de Falhas em Aplicativos

Para apoiar empresas de desenvolvimento no acompanhamento de possíveis falhas (*crashes)* em softwares implantados, normalmente são utilizadas ferramentas para gerar relatórios detalhados para apoiar a manutenção do software. Tais relatórios são gerados a partir de informações coletadas durante eventos de falha. Existem ferramentas, como o *Firebase Crashlytics* e *Sentry* para apoiar nesse processo, e que enviam esses relatórios para servidores e permitem que possam ser visualizados pelos desenvolvedores através de um *dashboard* (painel de controle) com interface amigável, conforme ilustrado pela Figura 1. Isso facilita a análise dessas informações já que um evento de falha pode ser capturado uma ou várias vezes em apenas um dispositivo onde o software está instalado ou em vários dispositivos. Dessa forma, fica fácil ver que o software está funcionando mal e entender em quais linhas de código pode estar o problema. Vale ressaltar que tais ferramentas não só capturam e enviam o erro (*crash*), mas também informações que podem ser muito úteis, como mensagens de alerta (*warnings*), de erros (*error*) e dados informativos (*infos*).

O serviço de monitorar *crashes* pode ser terceirizado, ou seja, pode-se utilizar uma ferramenta já existente no mercado ao invés de desenvolver uma ferramenta específica para a empresa ou para o software. Exemplos de ferramentas existentes para isso são o *Sentry* e *Firebase crashlytics*.

A ferramenta *Sentry* tem como objetivo monitorar, capturar falhas, registrar e organizar essas falhas por meio de informações mais detalhadas [*Sentry* 2022], como as que são mostradas na Figura 1.

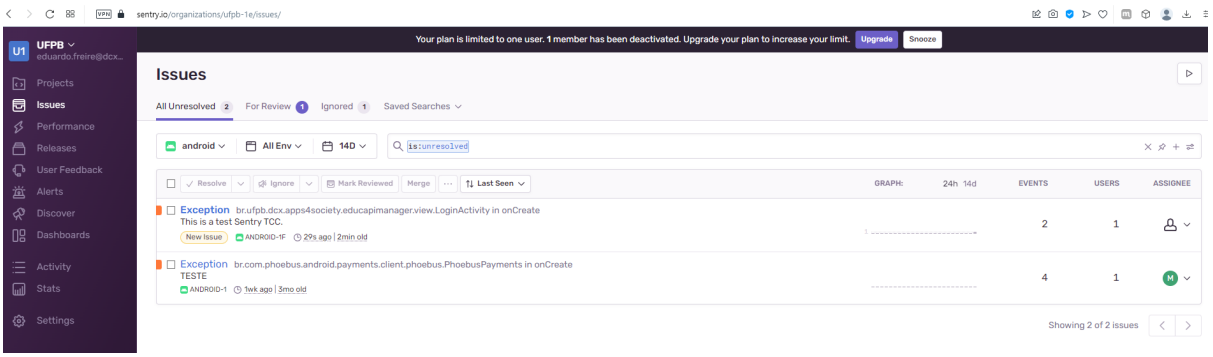

Figura 1. Tela de *Dashboard* da ferramenta *Sentry*

O funcionamento desta ferramenta acontece da seguinte maneira: no momento em que a exceção não tratada acontece, a ferramenta captura e grava um *backtrace*[7], isto é, uma lista de chamadas de função que estão ativas em uma *thread* (fluxo de execução). A ferramenta gera um relatório que dá detalhes sobre a falha. Após capturar esses detalhes, essas

[7] GNU: Disponível em: https://www.gnu.org/software/libc/manual/html_node/Backtraces.html/. Acesso em 04 Setembro de 2022

informações são gravadas em um arquivo no formato *JSON* e enviadas para um servidor remoto do *Sentry* assim que o aplicativo for executado novamente [*Sentry*[8]].

3. EducAPI e EducAPI-Manager

O sistema *EducAPI* é um serviço *Web* colaborativo construído com o propósito de facilitar a construção de diferentes objetos de aprendizagem para alfabetização e o gerenciamento das palavras e temas utilizados por estes [Ludgério et al. 2021]. Ele permite o gerenciamento de recursos multimídia por meio de um serviço cujo funcionamento geral está ilustrado pela Figura 2. Por meio dele, podem ser cadastrados desafios (palavras) e temas (contextos) por diferentes pessoas com imagens/vídeos/áudios correspondentes para que possam ser utilizados por diferentes aplicativos para alfabetização. A ferramenta desenvolvida para permitir este cadastro através de dispositivos móveis é o aplicativo EducAPI Manager, ilustrado pela Figura 3. O aplicativo está disponível no endereço "https://github.com/EduardoGhost/EducAPI-Manager-branch_edu/releases/tag/APK-new" e o seu código fonte está disponível em "https://github.com/EduardoGhost/EducAPI-Manager-branch_edu"

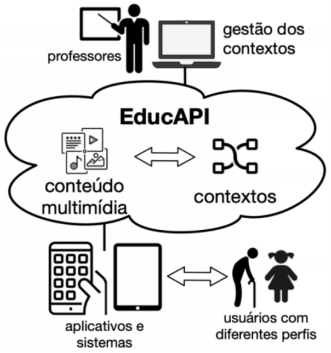

Figura 2: EducAPI. Fonte: Ludgério et. al (2021)

Por meio do aplicativo EducAPI-Manager, pode-se visualizar diferentes contextos cadastrados e cadastrar novos, além de incluir novos desafios e imagens associadas a estes desafios de forma que possam ser utilizados em diferentes aplicativos para alfabetização. Na Figura 3 são apresentadas três telas do aplicativo. Na tela à esquerda são mostrados alguns contextos cadastrados (Zoológico, Shopping e Floresta), bem como imagens que os representam. Na tela ilustrada à direita são mostrados desafios relacionados ao contexto "Floresta" como árvore, jacaré e pássaro.

[8] SENTRY. Disponível em: https://docs.sentry.io/platforms/android/. Acesso em 04 Setembro de 2022

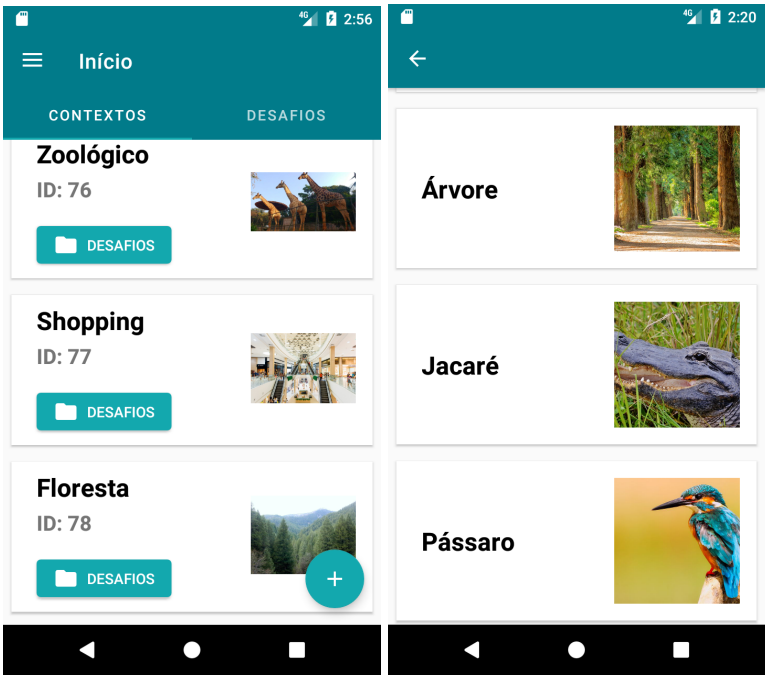

Figura 3: Telas do aplicativo EducAPI-Manager.

4. Metodologia

Inicialmente foi necessário se familiarizar com o tema abordado neste trabalho, tendo como base livros da engenharia de software como Pressman (2021) e Morais (2020). Este trabalho está relacionado à área manutenção de software e especificamente à área de manutenção corretiva. Com base em consultas na literatura com foco nessa área foi construído então o referencial teórico do trabalho.

No que diz respeito ao aplicativo EducAPI-Manager, foi necessário saber se o aplicativo era compatível com a ferramenta *Sentry*. Foi também necessário baixar o código fonte do aplicativo EducAPI-Manager do repositório remoto *GitHub*, no endereço "https://github.com/EduardoGhost/EducAPI-Manager-branch_edu". O *Android Studio* versão 2021.3.1 *beta* 2 foi o *framework* escolhido para manusear o código fonte do aplicativo e verificar seu funcionamento por meio de emuladores.

No que diz respeito à ferramenta *Sentry,* foi criada uma conta para poder utilizá-la no endereço eletrônico "https://sentry.io/". Através de um navegador qualquer pode-se acessar a interface principal desta ferramenta (*dashboard*) e também ter acesso à chave (*key*) que deverá ser referenciada no aplicativo por meio da configuração do arquivo *AndroidManifest*.

Com o ambiente pronto, foi possível experimentar a ferramenta no aplicativo e verificar possíveis falhas por meio de dois emuladores e de um dispositivo *Android*. Após a experiência, foram relatados os passos utilizados e as principais análises para apoiar outros desenvolvedores.

O aplicativo com a ferramenta previamente implementada foi instalado no dispositivo e nos emuladores, a fim de tentar simular o ambiente real, ou seja, o ambiente em que o usuário vai utilizar todas as funcionalidades do aplicativo normalmente. Com isso, pretendia-se monitorar e tentar capturar alguma falha do aplicativo.

O dispositivo para os testes foi um aparelho Moto G20, e os emuladores utilizados foram o *Galaxy Nexus API 25* e *Pixel 3a API 32*.

Foram exercitadas funcionalidades do aplicativo, ou seja, login de usuário, logout, sobre, contato, navegação sobre a listagem de contextos, desafios e opções do menu lateral, outras funcionalidades exercitadas foram o cadastro de desafios, exclusão de contextos e desafios, edição de contextos e desafios (alteração de imagens utilizadas e alteração das palavras utilizadas) e exibição de contextos e desafios. Ao executar essas ações foram encontrados problemas no início da aplicação, ao realizar *login*, e também por vezes ao exibir contextos cadastrados, conforme será detalhado a seguir.

Caso não fossem encontradas falhas naturais no aplicativo, seriam introduzidas falhas artificiais como por exemplo, ocupar toda a memória com muitos aplicativos abertos ou seriam feitas alterações no código fonte para induzir de forma intencional as falhas. Estas alterações seriam, por exemplo, a exclusão de algum tratamento de exceção ou a alteração em um comando condicional.

No que diz respeito a falha do aplicativo, se o usuário navegar sobre a tela listagem de contextos, o aplicativo pode parar de funcionar, e um erro de falha de requisição pode acontecer às vezes no aplicativo, basta tentar fazer o *login*.

5. Resultados

Nessa seção serão apresentados e discutidos os principais resultados deste trabalho com o intuito de apoiar outros desenvolvedores que necessitem utilizar ferramentas como o *Sentry*.

5.1. Passos para utilização do Sentry no aplicativo EducAPI-Manager

Nesta seção serão apresentados os passos necessários para a utilização do *Sentry* no aplicativo EducAPI-Manager.

Para utilizar o *Sentry* é necessário criar uma conta acessando o endereço eletrônico "https://sentry.io/", conforme ilustra a Figura 4.

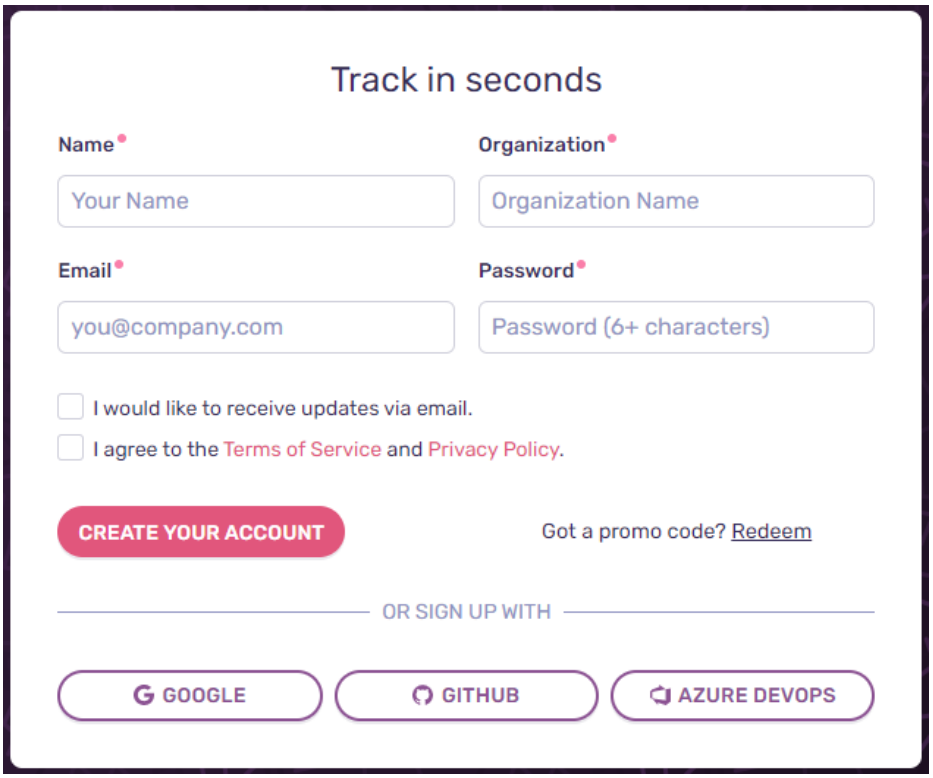

Figura 4. Tela de cadastro do Sentry

Após o login na plataforma, pode-se acessar a interface principal desta ferramenta (*dashboard*) e também ter acesso à chave (*key*), conforme ilustra a Figura 4. Esta chave deverá ser referenciada no aplicativo por meio da configuração do arquivo *AndroidManifest,* ilustrado pela Figura 5.

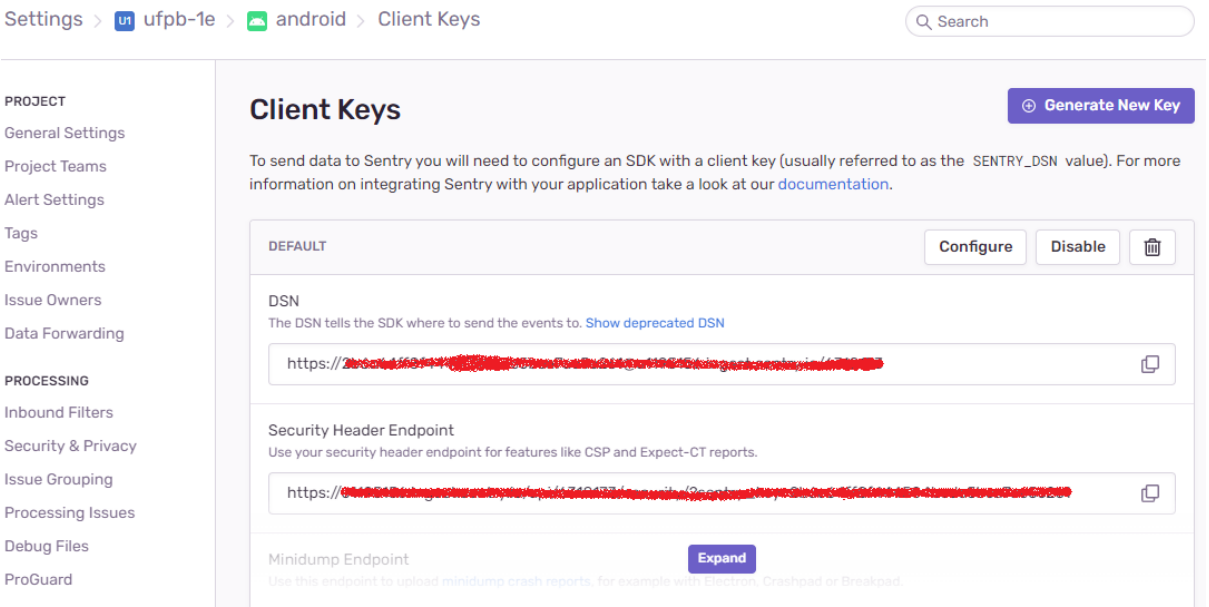

Figura 5. Tela da chave do cliente

No código fonte do aplicativo, acessado pelo *Android Studio*, no que diz respeito ao arquivo *build.gradle*, foi instalada a chave de acesso do *Sentry,* como pode ser visto na Figura 6. Essa chave de acesso vai permitir a conexão do aplicativo ao servidor do *Sentry*.

```
        </activity>
        <activity
            android:name=".view.NavDrawerActivity"
            android:label="EducAPI Manager" />

        <meta-data android:name="io.sentry.dsn"
            android:value="https://2b6c64ff8f444504bcde5bea7ae5e261@o1195156.ingest.sentry.io/6318173" />
    </application>

</manifest>
```

Figura 6. Ilustração do arquivo *AndroidManifest.xml* com a chave de acesso do *Sentry*

Para instalar o monitoramento no *Android* pelo *Sentry*, foi adicionado no arquivo *build.gradle* o texto indicado na Figura 7. Foi escolhida a versão 2.0.1 da ferramenta para não precisar alterar a versão do *gradle* do aplicativo.

```
    implementation 'io.sentry:sentry-android:2.0.1'
```

Figura 7. Referência à biblioteca do *Sentry*

Após esses procedimentos, o aplicativo foi instalado no dispositivo e emuladores citados anteriormente.

5.2. Análise de Falhas

Após instalar o aplicativo no dispositivo e nos emuladores, buscou-se inicialmente realizar o *login* por meio do aplicativo sendo executado para que o aplicativo EducAPI-Manager se conectasse ao serviço EducAPI. Não houve nenhuma falha nesse processo.

Posteriormente, buscou-se navegar pela tela de listagem de contextos e nesse momento foi constatado que o aplicativo parou de funcionar de forma inesperada tanto no dispositivo quanto nos emuladores. Ou seja, a aplicação falhou devido a uma falha grave (*crash*), como ilustra a Figura 8 em que é exibida a mensagem "*EducAPI Manager has stopped*." no emulador.

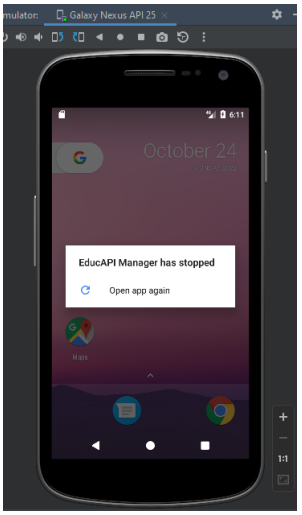

Figura 8: Registro do mau funcionamento aplicativo EducAPI-Manager.

As falhas verificadas foram capturadas pelo *Sentry* e enviadas ao servidor, onde podem ser visualizadas pelo painel (*dashboard)* da ferramenta. A Figura 9 mostra a tela com o resumo da falha, que foi do tipo "*IndexOutOfBoundsException*". Ao clicar nesta falha, pode-se ter acesso a um relatório detalhado da falha, como ilustra a Figura 10.

Moto G20

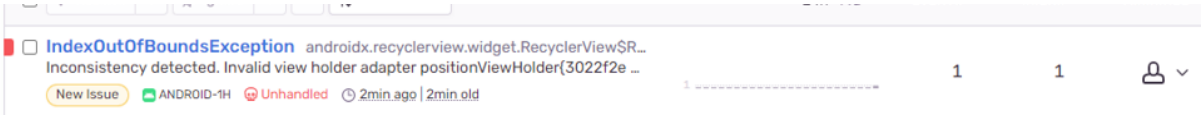

Figura 9: Tela do Dashboard do *Sentry* com a falha identificada

No relatório apresentado pelo *Sentry* é mostrado o caminho da execução até a falha em forma de *backtrace*, bem como informações do dispositivo no qual a falha foi capturada, conforme ilustra a Figura 9.

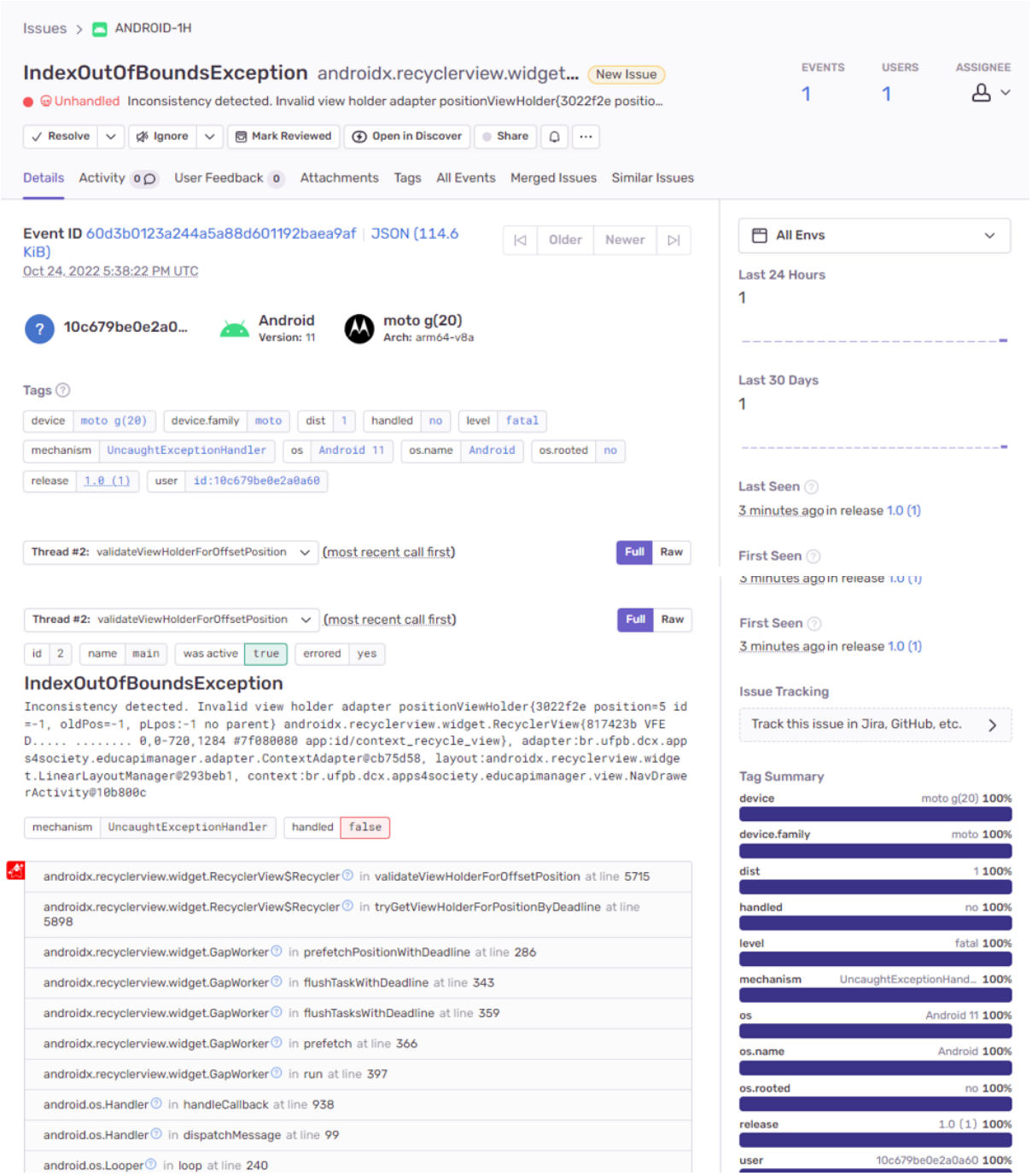

Figura 10: Tela detalhada do *Sentry* sobre a falha identificada

O *Sentry* classifica com cores diferentes cada problema de acordo com a falha que foi capturada, ou seja, usa-se vermelho para uma falha grave, como observado na Figura 9. Vale informar que existem mais cores de acordo com as implementações do desenvolvedor, como amarelo para *warnings* e azul para alguma informação que o desenvolvedor queira capturar. Além de classificar com cor vermelha uma falha grave, o *Sentry* coloca um ícone de caveirinha abaixo da falha ou informação capturada (ver Figura 9). Essas classificações são bem úteis no que diz respeito a visualização dessas questões.

A falha grave onde o aplicativo parou de funcionar foi registrada no *Sentry* com o título *IndexOutOfBoundsException*. Observando o *backtrace* desta falha, na visão de

desenvolvedor, pode-se observar que o problema está relacionado ao acesso a uma posição não válida de um *array*. Outra possibilidade, segundo pesquisas realizadas na internet, é algum problema no *recyclerview* do *Android* e que pode ser corrigido com uma implementação de tratamento de exceção, ou seja, bloco de *try/catch*.

Outras informações que podem ser muito úteis para o desenvolvedor ao analisar uma falha como o tipo de dispositivo, versão do sistema operacional, *release* do aplicativo, nível de bateria, espaço disponível na memória, versão do *kernel*, resolução do dispositivo, orientação da tela e linguagem são também mostradas no relatório do Sentry, conforme ilustra a Figura 10.

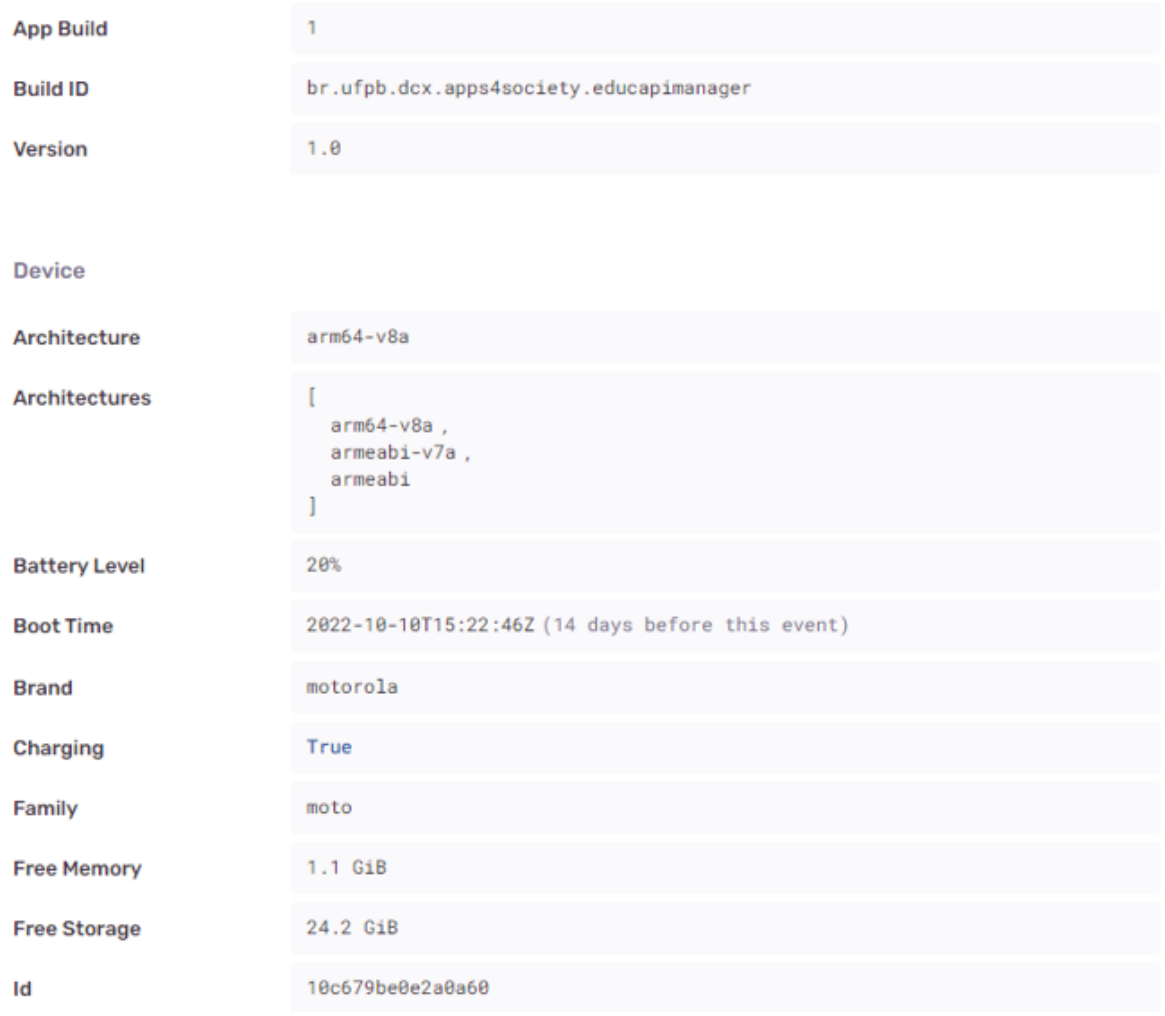

Figura 10: Tela do Sentry com mais detalhes sobre a falha identificada

Ao observar no *dashboard Sentry* as falhas obtidas através do uso do emulador, foram identificados dois problemas, conforme ilustra a Figura 11. Uma delas é igual à identificada no dispositivo (*IndexOutOfBoundsException*) e a outra, menos grave, pela cor laranja, mostrava que a aplicação não estava respondendo por mais de 4 segundos (*ApplicationNotResponding*). Pesquisando, viu-se que não se tratava de uma falha na aplicação, mas sim um erro de configuração do *Sentry* que considerou um tempo limite (*timeout)* insuficiente já que o emulador normalmente leva mais de 4 segundos para iniciar.

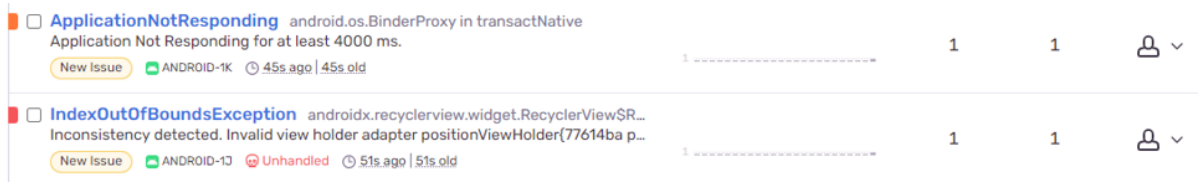

Figura 11: Tela do Sentry com as falhas identificadas utilizando o emulador Galaxy

Um detalhamento da falha de *IndexOutOfBoundsException* obtida no emulador está ilustrado na Figura 12.

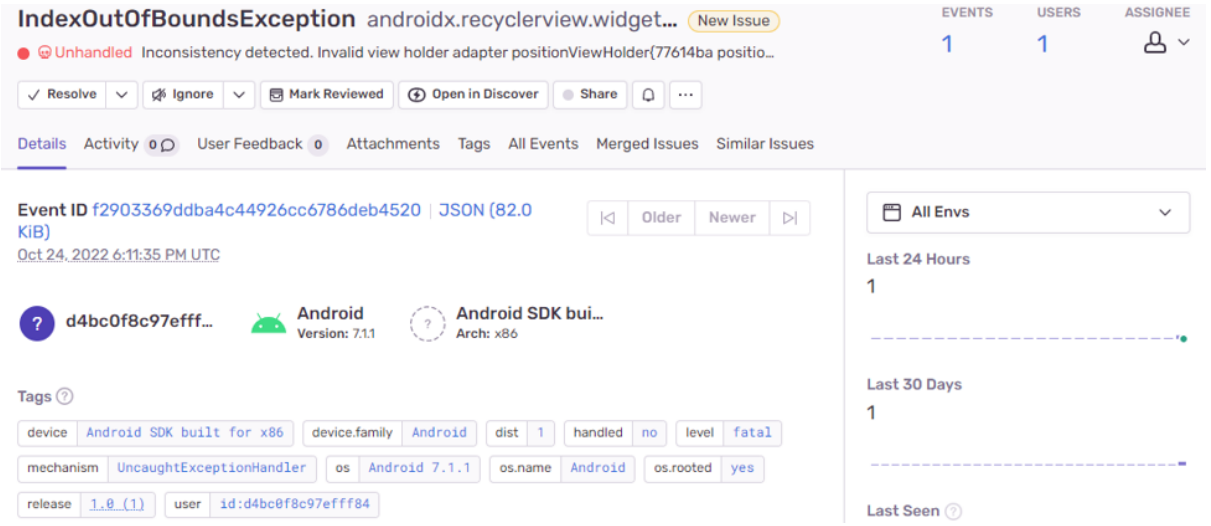

Figura 12: Tela do *Sentry* com o detalhamento de uma falha identificadas utilizando o emulador *Galaxy Nexus*

5.3. Uso do Sentry para expor falhas comuns encontradas por usuários

Utilizando o aplicativo, verificou-se que muitas vezes ao tentar realizar o *login*, era mostrada uma mensagem de "*Falha na requisição*", como pode ser visto na Figura 13. Vale ressaltar que este problema não acontecia sempre e considerou-se interessante investigá-lo melhor e utilizar o *Sentry* para monitorar a ocorrência desse problema, que muito provavelmente estava relacionado ao serviço e não ao aplicativo em si.

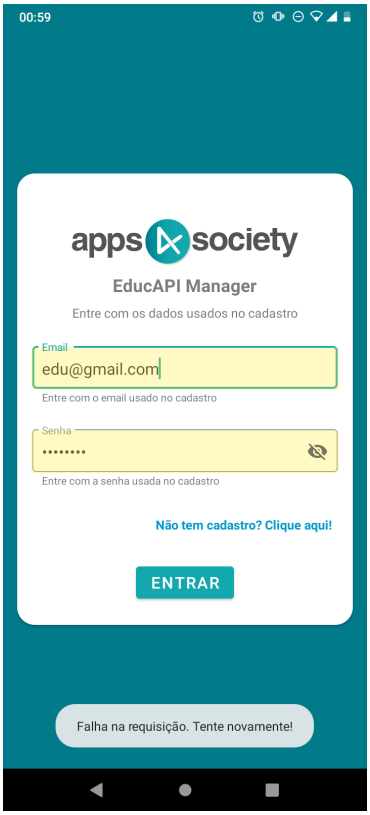

Figura 13: Imagem da falha de requisição na tela de *login*

Para monitorar o problema, um método do *Sentry* foi chamado (linha 97 da Figura 14) no código do aplicativo, especificamente no método onde a chamada de API é tratada caso aconteça algum problema de conexão (ou seja, caso o aplicativo não consiga se conectar ao serviço EducAPI).

```
LoginActivity.java
76    @Override
77    public void onResponse(Call<LoginResponse> call, Response<LoginResponse> response) {
78
79
80        if (response.code() == 200) {
81            LoginResponse loginResponse = response.body();
82            gerenteDeSessao.saveAuthToken(loginResponse.getToken());
83            userAuth = new UserDTO(userLogin.getEmail(), userLogin.getPassword());
84            CreateObjectFacade.Companion.getInstance().setTempSession(new Session(userAuth));
85            System.out.println(CreateObjectFacade.Companion.getInstance().getTempSession().getCreator().getEm
86            Toast.makeText(context, text: "Sessão iniciada",Toast.LENGTH_SHORT).show();
87            openNavDrawerActivity();
88
89        } else {
90            Toast.makeText( context: LoginActivity.this,  text: "E-mail ou senha incorretos! Verifique e tente nova
91        }
92    }
93
94    @Override
95    public void onFailure(Call<LoginResponse> call, Throwable t) {
96        Toast.makeText( context: LoginActivity.this,  text: "Falha na requisição. Tente novamente!", Toast.LENGTH_L
97            Sentry.captureException(t, LoginActivity.this);
98    }
```

Figura 14: Imagem do código inserido para capturar a falha de requisição. Fonte: O Autor

O aplicativo foi então instalado novamente no dispositivo e nos emuladores para observar a ocorrência do problema. Com isso o *Sentry* capturou e enviou o problema para o servidor de forma que pudesse ser visualizado pela *dashboard,* conforme ilustra a Figura 15.

O *Sentry* nomeou o problema com o título *SocketTimeOutExceptionn.* Na Figura 15, a primeira questão trata do problema do dispositivo Moto G20 e o emulador Pixel 3a e a segunda questão se referia à falha no emulador *Galaxy Nexus*. Apesar dessa separação, ao analisar, viu-se que o problema é o mesmo, ou seja, excedeu-se o tempo limite para conexão com servidor. Analisando esse aspecto, isso pode ser visto como um ponto de melhoria da ferramenta para facilitar para desenvolvedores a análise das falhas encontradas quando forem iguais.

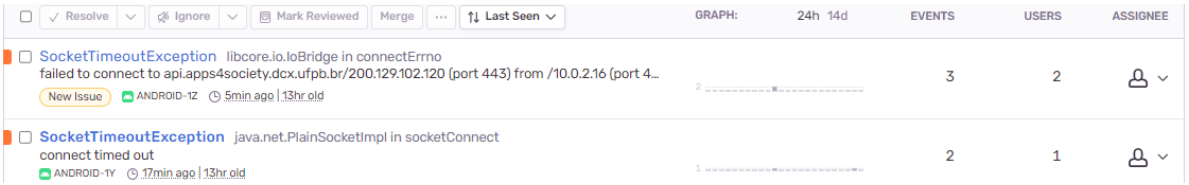

Figura 15 - Imagem da falha de requisição capturada

Como está ilustrado na Figura 16, o relatório diz que houve uma falha de conexão com o serviço e o tempo limite ao tentar conectar se excedeu, sendo assim foi lançada uma exceção.

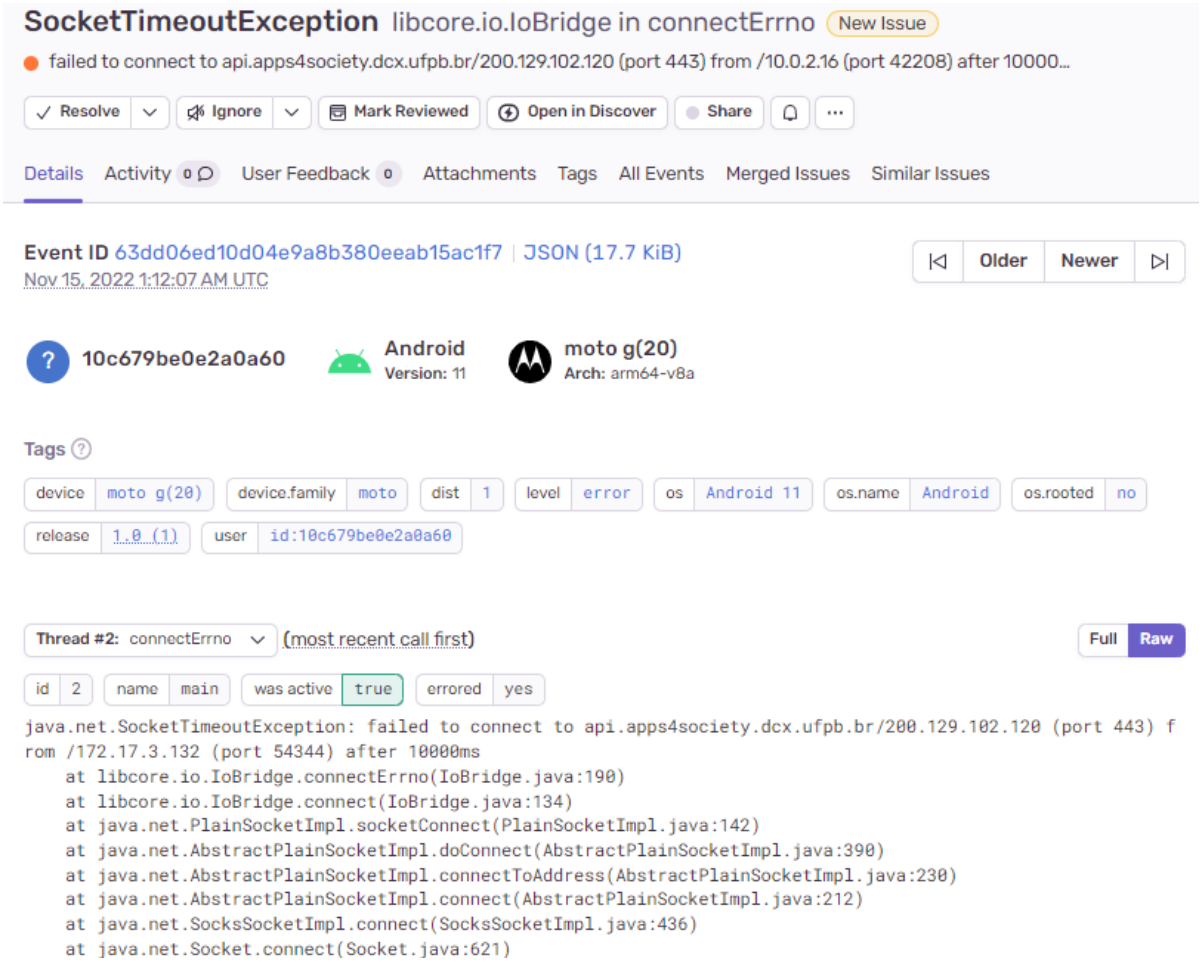

Figura 16 - Imagem do relatório da falha de requisição

Falhas encontradas	**Possíveis Soluções**
IndexOutOfBoundsException	Implementar um bloco de *try/catch*
ApplicationNotResponding	Falha da própria ferramenta por limite de tempo
Falha na requisição (erro)	Corrigir a API

Quadro 1 - Quadro ilustrativo das falhas

6. Conclusões e Trabalhos Futuros

Este relatório técnico teve como objetivo principal apoiar os desenvolvedores no que diz respeito a manutenção corretiva de software, explorando o uso de uma ferramenta de monitoração de falhas e relatando sua aplicação em um aplicativo *Android.* A ferramenta *Sentry* foi utilizada com sucesso no aplicativo EducAPI-Manager e foi capaz de capturar uma falha grave e que deve ser investigada. Considerando a experiência obtida, verificou-se que o *Sentry* foi capaz de capturar e registrar problemas encontrados na execução do aplicativo e que pode ser útil para melhoria da qualidade deste e de outros aplicativos.

Foram definidos os passos para a integração da ferramenta *Sentry* no aplicativo, como pode ser observado na Seção 5.1. Dessa forma qualquer desenvolvedor vai conseguir através deste passo a passo implementar a ferramenta em outro aplicativo *Android*.

Foram registradas as falhas capturadas e visualizadas pela *dashboard* do *Sentry*, e foram dadas explicações sobre informações que podem ser úteis ao desenvolvedor ao necessitar utilizar a ferramenta *Sentry* ou ferramenta similar.

A *dashboard* do *Sentry* tem uma interface amigável de fácil interação e visualização como pode ser visto na Figura 1 e na Figura 10.

No que diz respeito à usabilidade da ferramenta, o desenvolvedor pode ter uma certa dificuldade na instalação se ele não atentar para questões como a compatibilidade do *Gradle* com a versão da ferramenta.

Se o desenvolvedor desejar capturar alguma informação além das falhas, ele vai ter que implementar manualmente métodos do *Sentry* no código. Estes métodos podem ser visualizados na documentação da ferramenta. Alguns desenvolvedores podem ter uma certa dificuldade, pois a documentação está em inglês.

Foi mostrado como capturar um erro que pode acontecer às vezes no aplicativo e exibi-lo no *dashboard* do *Sentry*. Sendo assim, as principais contribuições deste trabalho foram então demonstrar a importância de monitorar aplicativos através de ferramentas próprias para isso, além de documentar um passo a passo do uso da ferramenta, demonstrando como capturar qualquer erro que pode eventualmente acontecer no aplicativo e analisar as falhas do aplicativo.

Como trabalhos futuros, pretende-se utilizar outros recursos disponíveis no *Sentry* no aplicativo e mostrar como podem ser úteis, como a captura de *warnings* e *infos*. É interessante também no futuro alterar a versão do *Gradle* utilizada no projeto do

EducAPI-Manager para que se possa explorar a versão 6.0.0 do *Sentry* ou outra posterior para analisar possíveis falhas do aplicativo, já que neste trabalho foi utilizada a versão 2.0.1.

Além disso, pretende-se no futuro utilizar a ferramenta em sistemas *web* e em outros aplicativos do projeto *Apps4Society* e ajudar assim na correção de possíveis defeitos encontrados. Outra proposta de trabalho futuro é investigar outras ferramentas similares como o *Crashlytics Firebase* e fazer um comparativo entre as ferramentas.

Referências

Android Developer. Disponível em: <https://developer.android.com/topic/performance/vitals/crash/> Acesso em: 03 ago. 2022.

Apps4Society. Construindo Aplicativos que Impactam Positivamente na Sociedade Disponível em: <https://apps4society.dcx.ufpb.br> Acesso em: 18 dez. 2022.

Firebase crashlytics. Disponível em: <https://firebase.google.com/docs/crashlytics> Acesso em: 03 set. 2022.

Gnu - Backtraces. Disponível em: <https://www.gnu.org/software/libc/manual/html_node/Backtraces.html> Acesso em: 04 set. 2022.

Gradle Build Tool. Disponível em: <https://gradle.org/> Acesso em: 03 ago. 2022.

Java - Java Oracle. Disponível em: <https://www.java.com/pt-BR/> Acesso em: 07 set. 2022.

Kotlin - A modern programming language that makes developers happier. Disponível em: <https://kotlinlang.org> Acesso em: 07 set. 2022.

Ludgerio, R.; Amorim, L. V. ; Dantas, Emerson; Rebouças, Ayla Dantas (2021). O sistema colaborativo EducAPI e sua avaliação como ferramenta para apoiar a alfabetização. Revista Tecnologias na Educação, v. 35, p. 1-18.

Morais, Izabelly Soares D.; Zanin, Aline (2020). **Engenharia de software**. Grupo A. E-book. ISBN: 9788595022539. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788595022539/. Acesso em: 03 set. 2022.

Pressman, Roger S.; Maxim, Bruce R. (2021) **Engenharia de software.** Grupo A. ISBN: 9786558040118. E-book. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9786558040118/. Acesso em: 14 ago. 2022.

Rebouças, Ayla Débora Dantas de Souza (2010). Aumento a confiança nos resultados de testes de sistemas multi-threaded: evitando asserções antecipadas e tardias. Tese de Doutorado. Universidade Federal de Campina Grande — Campina Grande-PB.

Sentry. Disponível em: <https://sentry.io/> Acesso em: 03 ago. 2022.

Statista - Mobile Android operating system market share by version worldwide from January 2018 to August 2022 - 2022. Disponível em: <https://www.statista.com/statistics/921152/mobile-android-version-share-worldwide/> Acesso em: 28 out. 2022.