UNIVERSIDADE ESTADUAL DE FEIRA DE SANTANA
Autorizada pelo Decreto Federal nº 77.496 de 27/04/76
Recredenciamento pelo Decreto nº17.228 de 25/11/2016
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
COORDENAÇÃO DE INICIAÇÃO CIENTÍFICA

XXVIII SEMINÁRIO DE INICIAÇÃO CIENTÍFICA DA UEFS
SEMANA NACIONAL DE CIÊNCIA E TECNOLOGIA - 2024

ELECTRIC EYE: FERRAMENTA ON-LINE PARA AUTOMATIZAÇÃO DE CRIAÇÃO DE SISTEMAS DE GERENCIAMENTO DE EMERGÊNCIAS URBANAS

Ian Zaque Pereira de Jesus dos Santos[1]; Thiago Cerqueira de Jesus[2];

1. Bolsista PIBIC/CNPq, Graduando em Engenharia de Computação, Universidade Estadual de Feira de Santana, e-mail: ianzaque.uefs@gmail.com

2. Orientador, Departamento de Tecnologia, Universidade Estadual de Feira de Santana, e-mail: tcjesus@uefs.br

PALAVRAS-CHAVE: Cidades inteligentes, Detecção de emergências, Automatização de sistemas.

INTRODUÇÃO

A urbanização dos centros urbanos e áreas adjacentes traz adversidades que ameaçam a população e o meio ambiente, como emergências urbanas e desastres naturais, podendo resultar em grandes catástrofes. Em cidades inteligentes, sistemas de detecção de emergências permitem localizar rapidamente essas situações, evitando maiores danos. No entanto, para isso, é essencial que as cidades estejam preparadas com ferramentas adequadas para agilizar processos e solucionar gargalos.

A integração de tecnologias de sensoriamento remoto é crucial para garantir a gestão eficiente de recursos e a tomada de decisões em situações emergenciais. O sensoriamento remoto em rede é fundamental para implementar sistemas de detecção de emergências (SDE). Assim, este trabalho propõe uma plataforma on-line automatizadora para facilitar a implantação e configuração de sensores, otimizando o gerenciamento de emergências urbanas.

MATERIAL E MÉTODOS OU METODOLOGIA (ou equivalente)

Tendo em vista que o objetivo deste trabalho é a descrição da implementação de uma plataforma web para automatização de sistema de detecção de emergências (SDE), foi necessário seguir uma abordagem incremental, prática e iterativa. A plataforma *Electric Eye* foi inteiramente baseada no SDE desenvolvido por Coelho em [4], portanto utilizou nomenclaturas, siglas, arquiteturas, padrões e políticas de comunicação para casamento e alinhamento completo de funcionalidades com o que foi ali descrito e implementado. Além disso, o Electric Eye pode se comunicar com o SDE para troca de dados, criação e manipulação de entidades e seus dados por meio do protocolo MQTT quando conectado ao mesmo broker. Foi realizada uma revisão bibliográfica em cima de artigos, sites oficiais de determinadas tecnologias e documentações que proporcionou o entendimento das temáticas abordadas relacionadas ao SDE e a plataforma *Electric Eye*.

Um dos primeiros passos no desenvolvimento de um software é a definição de requisitos funcionais e não funcionais dela. Foram levantados 20 requisitos funcionais (RF) e 20

não funcionais (NF), estes últimos agrupados a partir de categorias, tais como usabilidade, confiabilidade, desempenho, segurança, distribuição e padrões, para melhor classificação. Os principais requisitos estão descritos resumidamente na tabela a seguir e podem ser visualizados completamente no repositório [17].

Tipo	Código	Nome
Funcional	RF001	Acesso a telas de recursos
Funcional	RF012	Criação de contas de usuário
Funcional	RF020	Envio de dados para tópicos MQTT
Não funcional	NF001	Acesso web
Não funcional	NF003	Indisponibilidade do Sistema
Não funcional	NF004	Notificação de erros
Não funcional	NF009	Autenticação no sistema
Não funcional	NF016	Arquitetura de software
Não funcional	NF019	Consumo de dados

Tabela 1: Principais requisitos do sistema. Fonte: Próprio Autor.

Com os requisitos planejados e baseados no SDE [4], a arquitetura foi dividida em três módulos:

1. *Front-end*: Interface de usuário (UI) responsável pelas funções de CRUD. O usuário interage com a aplicação, gerenciando cadastros, entidades e verificando a integração com o SDE, atuando como cliente do Electric Eye.
2. API: Facilita a comunicação entre cliente e servidor, controlando o fluxo de mensagens.
3. *Back-end*: Trata as requisições da API, gerencia entidades, regras de negócio e a conexão com o banco de dados, gerando as páginas para o front-end, atuando como servidor.

Com a arquitetura definida, as tecnologias foram escolhidas. O Vue.js foi usado no *front-end* por sua versatilidade e escalabilidade [18]. O Axios, uma biblioteca isomórfica baseada em promessas HTTP, foi escolhido para a API. No *back-end*, o Laravel 7 foi selecionado por sua integração com Vue.js, robustez e amplo suporte [20]. A comunicação entre o *Electric Eye* e o sistema de detecção de emergências (SDE) ocorre exclusivamente via protocolo MQTT. As telas foram baseadas no padrão do Laravel 7, com visual definido pelo Bootstrap. A prototipação foi feita no Figma, auxiliando na padronização e visualização do design.

RESULTADOS E/OU DISCUSSÃO (ou Análise e discussão dos resultados)

A implementação das telas seguiu a prototipação, cadastro de entidades e modelagem do banco de dados. O acesso ao sistema requer cadastro e verificação de e-mail. Cada entidade tem um controller que gerencia suas ações, permitindo ao usuário autorizado executá-las via interface (UI) e receber um alerta de status. Ao cadastrar uma UDE, uma mensagem com seu endereço MAC é enviada ao tópico *subscribe* do broker MQTT do SDE. Em seguida, a tabela de UDEs atualizada é enviada ao tópico *update_node_table*,

e a configuração da UDE, incluindo sensores, zona de interesse e parâmetros de emergência, é enviada ao tópico *config*.

O primeiro objetivo específico (Projetar e desenvolver a ferramenta on-line) foi cumprido ao desenvolver uma plataforma web de fácil instalação em servidor e compreensão de utilização pelo usuário e totalmente customizável à qualquer região ou cidade que queira modelar o SDE em conjunto com o *Electric Eye*. Este objetivo foi cumprido parcialmente pois não abrange a definição de eventos críticos. Como ainda não há registro dos eventos críticos na plataforma, não foi possível implementar algoritmos de detecção de padrões a partir dos dados coletados. Portanto, não foi possível cumprir o segundo objetivo específico: implementar algoritmos de detecção.

O terceiro objetivo específico (integrar tecnologias de comunicação e rede) foi cumprido completamente. Ao instalar a plataforma *Electric Eye* em um servidor de produção e estando o *broker* do SDE online e operacional, pode-se manter uma conexão e comunicação com este por meio da troca de mensagens nos tópicos. O *software* dispõe da biblioteca *php-mqtt/laravel-client* que permite que se conecte por um cliente autorizado ao *broker*. Assim, o objetivo foi atendido totalmente, pois possibilita que o SDE e as unidades de detecção se comuniquem com a plataforma em tempo real independentemente de distâncias entre os itens, bastando apenas ter conexão à internet. O quarto objetivo específico também foi totalmente cumprido ao ter desenvolvido uma interface de usuário de fácil compreensão e uso para que possa ser coerente e transparente. As telas, em sua maioria, são compostas pela barra lateral com acesso às ações do sistema e pela tabela de listagem de itens cadastrados. Na tabela há as informações pertinentes, apresentando a entidade de forma resumida e simples, exibindo também as opções de edição e deleção de itens.

CONSIDERAÇÕES FINAIS (ou Conclusão)

A plataforma *Electric Eye* tem como objetivo automatizar e gerenciar entidades que integram um sistema de detecção de emergências, modelando e flexibilizando-o de modo que centralize o monitoramento de emergências e variáveis personalizadas por meio de uma interface intuitiva, amigável e de fácil utilização. Ao cumprir parcialmente os objetivos pode-se notar e ressaltar o conjunto de conhecimentos que foram trabalhados que denotam à várias áreas da computação, abrangendo desde redes, engenharia de *software*, arquitetura de computadores, sistemas digitais, entre outros. O trabalho também ressalta a importância de um design centrado no usuário, com uma interface que facilite a interação e o controle do sistema, permitindo que os gestores possam agir de maneira eficaz diante de cenários críticos.

REFERÊNCIAS

[1] JESUS, C.; PORTUGAL, P.; COSTA, D. G.; VASQUES, F. Reliability and detectability of emergency management systems in smart cities under common cause failures. *Sensors*, v. 24, n. 9, 2024.

[2] YAZAR, U. T.; AKSIT, M. An architectural framework for the allocation resources in emergency management systems. In: TEKINERDOGAN, B.; AKSIT, M.; CATAL,

C.; HURST, W.; ALSKAIF, T. (Eds.). *Management and Engineering of Critical Infrastructures*. Academic Press, 2024. p. 223-241.
[3] SHAH, S. A.; SEKER, D. Z.; RATHORE, M. M.; HAMEED, S.; BEN YAHIA, S.; DRAHEIM, D. Towards disaster resilient smart cities: Can internet of things and big data analytics be the game changers? *IEEE Access*, v. 7, 2019.
[4] COELHO, G. A. A.; JESUS, T. C.; COSTA, D. G. Urban emergency detection system using hierarchical, collaborative and configurable wireless sensor networks. In: *2023 XIII Brazilian Symposium on Computing Systems Engineering (SBESC)*, 2023.
[5] COSTA, D. G.; VASQUES, F.; PORTUGAL, P.; AGUIAR, A. A distributed multi-tier emergency alerting system exploiting sensors-based event detection to support smart city applications. *Sensors*, v. 20, 2019.
[6] PENG, T.; KE, W. Urban fire emergency management based on big data intelligent processing system and internet of things. *Optik*, v. 273, p. 170433, 2023.
[7] GARCÍA, C. G.; MEANA-LLORIÁN, D.; LOVELLE, J. M. C. et al. A review about smart objects, sensors, and actuators. *International Journal of Interactive Multimedia & Artificial Intelligence*, v. 4, n. 3, 2017.
[8] JAVAID, M.; HALEEM, A.; RAB, S.; SINGH, R. P.; SUMAN, R. Sensors for daily life: A review. *Sensors International*, v. 2, p. 100121, 2021.
[9] RANI, S.; KUMAR, R. Bibliometric review of actuators: Key automation technology in a smart city framework. *Materials Today: Proceedings*, v. 60, p. 1800-1807, 2022.
[10] BALAJI, S.; NATHANI, K.; SANTHAKUMAR, R. Iot technology, applications and challenges: a contemporary survey. *Wireless Personal Communications*, v. 108, p. 363-388, 2019.
[11] MOUHA, R. A. R. A. et al. Internet of things (IoT). *Journal of Data Analysis and Information Processing*, v. 9, n. 2, p. 77, 2021.
[12] MQTT.org. Mqtt: The standard for iot messaging. Disponível em: https://mqtt.org. Acesso em: 23 jul. 2024.
[13] HiveMQ. Mqtt essentials. Disponível em: https://www.hivemq.com/mqtt. Acesso em: 23 jul. 2024.
[14] MISHRA, B.; KERTESZ, A. The use of mqtt in m2m and iot systems: A survey. *IEEE Access*, v. 8, p. 201071-201086, 2020.
[15] AMAZON Web Services. O que é uma api restful? Disponível em: https://aws.amazon.com/pt/what-is/restful-api. Acesso em: 23 jul. 2024.
[16] IBM. O que é uma api rest? Disponível em: https://www.ibm.com/br-pt/topics/rest-apis. Acesso em: 23 jul. 2024.
[17] SANTOS, I. Z. P. de J. Repositório - electric eye. Disponível em: https://github.com/ian-zaque/eye-of-the-beholder/blob/main/Requisitos%20de%20Software%20-%20Electric%20Eye%20-%20TCC%20I.pdf. Acesso em: 23 jul. 2024.
[18] YOU, E. Vue.js: The progressive javascript framework. Disponível em: https://vuejs.org. Acesso em: 23 jul. 2024.
[19] ZABRISKIE, J. J. S. M. Axios: Promise based http client for the browser and node.js. Disponível em: https://axios-http.com/docs/intro. Acesso em: 23 jul. 2024.
[20] LARAVEL. Laravel: The php framework for web artisans. Disponível em: https://laravel.com. Acesso em: 23 jul. 2024.