UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA DE TELEINFORMÁTICA

CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

GUSTAVO CIGNACHI

DESENVOLVIMENTO DE INTERFACE GRÁFICA PARA SISTEMA DE MONITORAMENTO E GESTÃO DE CAMPUS UNIVERSITÁRIO INTELIGENTE

FORTALEZA

2023

GUSTAVO CIGNACHI

DESENVOLVIMENTO DE INTERFACE GRÁFICA PARA SISTEMA DE MONITORAMENTO E GESTÃO DE CAMPUS UNIVERSITÁRIO INTELIGENTE

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Paulo Antonio Leal Rego

FORTALEZA

2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

C512d Cignachi, Gustavo.
Desenvolvimento de interface gráfica para sistema de monitoramento e gestão de Campus Universitário Inteligente / Gustavo Cignachi. – 2023.
79 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Tecnologia, Curso de Engenharia de Computação, Fortaleza, 2023.
Orientação: Prof. Dr. Paulo Antonio Leal Rego.

1. Campus Universitário Inteligente. . 2. Interface de Usuário. 3. Internet das Coisas. I. Título.
CDD 621.39

GUSTAVO CIGNACHI

DESENVOLVIMENTO DE INTERFACE GRÁFICA PARA SISTEMA DE MONITORAMENTO E GESTÃO DE CAMPUS UNIVERSITÁRIO INTELIGENTE

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Paulo Antonio Leal Rego (Orientador)
Universidade Federal do Ceará - UFC

Prof. Dr. Michel Sales Bonfim
Universidade Federal do Ceará - UFC

Prof. Me. Marcos Dantas Ortiz
Universidade Federal do Ceará - UFC

Aos meus pais, por acreditarem e terem me dado a chance de chegar até aqui. Mãe, Pai, o amor e o apoio de vocês foi o que por muitas vezes me motivou a seguir em frente.

AGRADECIMENTOS

Agradeço primeiramente a meus pais, por todo apoio e todos os sacrifícios que fizeram para que eu pudesse chegar até aqui. Agradeço a Juliana, minha companheira que esteve ao meu lado durante toda a elaboração deste trabalho, sem o apoio e a paciência dela não teria conseguido chegar até aqui. Queria agradecer ao Dr. Paulo Antonio Leal Rego pela oportunidade de poder atuar neste trabalho e pela paciência e orientação nestes últimos meses. Gostaria de agradecer também aos meus amigos que fiz dentro da Universidade Federal do Ceará, nossa amizade e convivência foi de extrema importância para mim durante todos esses anos. Por fim, gostaria de agradecer ao Bartholomew, meu companheiro diário que sempre esteve ao meu lado nos últimos cinco anos, sempre lembraremos de você e sentiremos sua falta.

“Todo mundo tem coisas que consegue e não consegue fazer. Eu vou fazer as coisas que você não consegue, e você faz as coisas que eu não consigo.”

(Sanji)

RESUMO

Caracterizada como uma revolução tecnológica, a Internet das Coisas (IoT) possibilita o desenvolvimento do *Smart Campus*, um modelo de campus universitário inteligente que busca melhorar a qualidade de vida na universidade e contribuir com a gestão de recursos, espaços e equipamentos, além da otimização de custos e tempo. No entanto, o *Smart Campus* tem como desafio o grande volume e a variedade de dados gerados, contexto em que é imprescindível buscar a melhor forma de visualizar essas informações. Logo, o objetivo geral deste trabalho é desenvolver uma interface gráfica para apresentação da integração de dados dos sensores e controle de atuadores e câmeras do sistema de monitoramento e gestão do Campus Inteligente da Universidade Federal do Ceará (UFC). A interface foi desenvolvida utilizando o framework *JavaScript React.js* e permite interagir remotamente com os dispositivos instalados no campus e acessar seus respectivos dados, além de selecionar medições específicas. Em testes de desempenho realizados nas telas da aplicação, os resultados apontam que o desempenho da interface é satisfatório, apresentando a necessidade de ajustes pontuais que podem ser realizados em trabalhos futuros a fim de melhorar ainda mais a experiência do usuário.

Palavras-chave: Campus Universitário Inteligente. Internet das Coisas. Interface de Usuário

ABSTRACT

Characterized as a technological revolution, the Internet of Things (IoT) allows the development of the Smart Campus, a model that seeks to improve the quality of life in a university and contributes to its management of resources, common spaces and equipment, while also optimizing costs and time. However, the Smart Campus presents the challenge of having a great volume and variety of generated data, making it essential to seek better ways of visualizing this information. Therefore, the general objective of this work is the development of a graphical interface to display the integration of sensor data and control of actuators and cameras of the monitoring system and management of the Smart Campus of the Federal University of Ceará (UFC). The interface was developed using React.js, a JavaScript framework, and allows the interaction with devices from the campus and their respective associated data, while also allowing the usage of specific visualizations. After running performance tests on the application, the results show that the interface's performance is satisfactory, needing small adjustments that can be performed in future works seeking to further improve the user's experience.

Keywords: Smart Campus. Internet of Things. User Interface

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

IoT	Internet das Coisas
TICs	Tecnologias da Informação e Comunicação
API	Interface de Programação de Aplicação
HSC	Human Smart Cities
ONU	Organização das Nações Unidas
SCI	Smart City Index
IMD	International Institute for Management Development
IA	Inteligência Artificial
GE	Generic Enabler
NGSI	Next Generation Service Interface
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
MQTT	MQ Telemetry Transport
ML	Machine Learning
UI	User Interface
PYPL	PopularitY of Programming Language
DOM	Document Object Model
MVC	Model-View-Controller
REST	Transferência Representacional de Estado
OSM	OpenStreetMap
ODS	Objetivos de Desenvolvimento Sustentável
SPA	Single Page Application

AWS	Amazon Web Services
FCP	First Contentful Paint
LGC	Largest Contentful Paint
TBT	Total Blocking Time
CLS	Cumulative Layout Shift
SP	Speed Index
s	Segundo
ms	Milisegundo
kB	Kilobyte
MB	Megabyte

SUMÁRIO

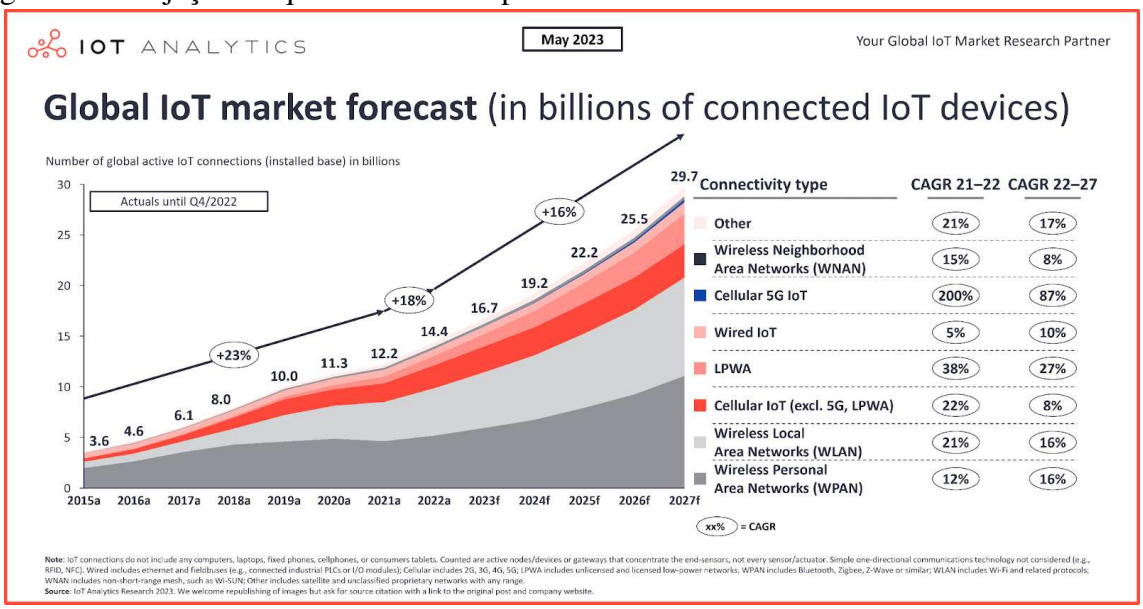
1 INTRODUÇÃO

Em novembro de 2022, a Organização das Nações Unidas (ONU) anunciou que a Terra atingiu a marca de 8 bilhões de habitantes. Para 2030, a projeção é de 8,5 bilhões de habitantes, 9,7 bilhões em 2050 e 10,4 bilhões em 2100 (BBCNEWS, 2022). No Brasil, o Instituto Brasileiro de Geografia e Estatística (IBGE) aponta que o país já alcançou uma população com cerca de 207,8 milhões de habitantes (MARTINI; ROSAS, 2022).

Nesse contexto, observa-se que atualmente as cidades já enfrentam fortes transformações sociais, econômicas e ambientais, "como o aumento da desigualdade social, acesso limitado aos serviços públicos básicos, problemas de mobilidade urbana e segurança" (NEVES *et al.*, 2017). Logo, em um futuro de crescimento populacional tão acentuado, esses problemas se tornam ainda mais desafiadores.

Dessa forma, a tecnologia pode atuar como uma aliada da vida nas cidades, com destaque para a *Internet of Things* (IoT), ou Internet das Coisas, uma rede que conecta à Internet diferentes tipos de dispositivos inteligentes que conseguem captar, transmitir e receber dados de forma automatizada. Com crescimento expressivo, até 2027 estima-se que no mundo existirão 29,7 bilhões de dispositivos IoT conectados, conforme a Figura 1 ilustra. Atualmente, esse número é de 16,7 bilhões de dispositivos (SINHA, 2023).

Figura 1 – Projeção da quantidade de dispositivos IoT instalados ate 2027



Fonte: (SINHA, 2023)

No contexto das cidades, a IoT está diretamente ligada ao conceito de *Smart City*, um modelo de cidade inteligente, inovadora, conectada e sustentável. Dessa forma, uma *Smart City* usa Tecnologias da Informação e Comunicação (TICs) e outros meios para melhorar a qualidade de vida, a eficiência da operação e dos serviços urbanos e a competitividade, buscando atender às necessidades das gerações presentes e futuras nos aspectos econômico, social, ambiental e cultural (NEGREIROS *et al.*, 2020).

Nesse contexto, pode-se citar o famoso caso da cidade espanhola Santander, que se tornou um grande laboratório para pesquisa e experimentação de arquiteturas, tecnologias facilitadoras essenciais, serviços e aplicativos para a Internet das Coisas no contexto das cidades inteligentes (CHENG *et al.*, 2015). Ao todo, são mais de 10 mil sensores espalhados pela cidade, captando dados em tempo real sobre luz, temperatura, ruído, nível de CO2, ocupação de estacionamentos, entre outros.

Derivado da *Smart City*, há também o *Smart Campus*, ou campus inteligente. Ambos são conceitos interligados, pois, geralmente, estão em um contexto socioeconômico, ambiental e geográfico similar, compartilhando infraestruturas, canais de comunicação, serviços, redes de transporte, desafios e necessidades semelhantes (DONG *et al.*, 2020). Assim, um campus inteligente pode, inclusive, servir como um laboratório para o surgimento de inovações a serem replicadas nas cidades inteligentes.

Em um projeto de *Smart Campus*, são diversos os tipos de dispositivos instalados e os dados que podem ser captados por eles. Por meio de câmeras, por exemplo, é possível identificar o nível de ocupação de estacionamentos e salas, enquanto outros dispositivos possibilitam monitorar temperatura, umidade e qualidade do ar. Além disso, com a captação de dados relacionados ao consumo de água e energia, é possível, inclusive, proporcionar uma melhor gestão desses recursos, evitando desperdícios e possíveis aumentos de custos.

Assim, o campus inteligente apresenta vantagens consideráveis, como a melhor gestão de recursos, espaços e equipamentos, além da otimização de custos e tempo (ABUARQOUB *et al.*, 2017). Outro fator positivo é o monitoramento e a preservação do meio ambiente, onde atualmente já é possível, por exemplo, evitar incêndios por meio do uso de sensores de temperatura (PALMA *et al.*, 2014). Assim, o campus inteligente permite não apenas uma melhor qualidade de vida, mas também um cenário mais sustentável.

No entanto, o modelo de campus inteligente enfrenta também desafios, como a numerosa quantidade de dispositivos necessários para cobrir a estrutura de um campus, bem

como o grande volume e a variedade de dados e informações por eles captados (ABUARQOUB *et al.*, 2017). Dessa forma, um importante fator a ser considerado nesse contexto é a visualização desse alto volume de dados gerado.

Assim, este trabalho tem como foco o projeto do Campus Inteligente da Universidade Federal do Ceará (UFC), que foi construído integrando infraestrutura, dados e pesquisadores de dois campi da universidade: Pici e Quixadá. A estrutura atual do projeto possui uma rede de sensores, atuadores e câmeras de monitoramento que se liga a um middleware FIWARE[1], além de possuir uma estrutura de unificação e monitoramento utilizando o Grafana. No entanto, devido a quantidade de dados envolvidos no monitoramento do Campus da UFC e as limitações de sua estrutura e customização, o Grafana se mostra uma ferramenta não muito favorável para a visualizações de dispositivos e métricas do campus.

1.1 Objetivo

Nesse contexto, o objetivo geral deste trabalho é desenvolver uma interface gráfica utilizando o framework *JavaScript React.js*[2] para apresentação da integração de dados dos sensores e controle de atuadores e câmeras do sistema de monitoramento do campus inteligente da UFC. O trabalho, então, busca apresentar uma interface de fácil uso e visualização, permitindo aos usuários acessar remotamente os dados de dispositivos de qualquer sala, bloco e campus; selecionar as medições feitas dentro de um intervalo de tempo específico e interagir de modo mais fácil e intuitivo com os dispositivos presentes.

Assim, este trabalho busca cumprir também quatro objetivos específicos a fim de atingir o objetivo geral de desenvolver a interface gráfica para o campus inteligente da UFC, sendo eles:

1. Definir e elencar as funcionalidades necessárias da aplicação;
2. Realizar a integração da aplicação com os dados do FIWARE e da API;
3. Desenvolver as visualizações necessárias a fim de implementar as funcionalidades definidas;
4. Realizar experimentos para avaliar a performance das funcionalidades desenvolvidas.

Dessa forma, a interface gráfica desenvolvida neste trabalho busca possibilitar não

[1] https://www.fiware.org/
[2] https://react.dev/

apenas um melhor acesso remoto aos dados obtidos com o sensoriamento inteligente do campus, mas também uma melhor experiência de visualização por parte dos usuários da aplicação, que a princípio são servidores e alunos da própria universidade.

Por fim, este trabalho está organizado com a seguinte estrutura: o Capítulo 2 apresenta a fundamentação teórica; no Capítulo 3 são apresentados os trabalhos relacionados e uma análise comparativa; no Capítulo 4 é apresentada a metodologia; no Capítulo 5 estão os resultados alcançados com o trabalho, além de uma análise de desempenho; e, por fim, o Capítulo 6 apresenta as considerações finais e as sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são apresentados conceitos e tecnologias fundamentais para o entendimento e desenvolvimento deste projeto.

A Seção 2.1 introduz e desenvolve conceitos relacionados à Internet das Coisas (IoT). A Seção 2.1.1 introduz conceitos relacionados a *Smart City*. A Seção 2.1.2 introduz conceitos relacionados a *Smart Campus*. A Seção 2.2 explica e elabora sobre a estrutura e desenvolvimento do *Smart Campus* em estudo neste trabalho, o Campus Inteligente da Universidade Federal do Ceará (UFC). Por fim, a Seção 2.3 entra nos conceitos e tecnologias envolvidas no desenvolvimento de uma interface de usuário para um *Smart Campus*.

2.1 Internet das Coisas

Com o passar do tempo e o desenvolvimento da sociedade, percebe-se o surgimento de novos avanços e conquistas, bem como desafios a serem superados em diferentes esferas. A tecnologia, então, vem acompanhando essa passagem do tempo e se tornando cada vez mais presente na vida das pessoas, seja por meio de smartphones, computadores ou mesmo ferramentas de Inteligência Artificial (IA).

Com isso, é perceptível que a tecnologia atua como um meio de tornar a vida real ainda mais conectada, dinâmica e prática, propondo avanços notáveis nas mais diversas áreas, como medicina, mobilidade, agricultura e educação. É nesse contexto que se nota a relevância e ascensão da Internet das Coisas, ou Internet of Things (IoT), termo usado pela primeira vez pelo pesquisador Kevin Ashton, em 1999.

Não há uma única forma de definir IoT, havendo discordâncias mesmo entre os pesquisadores e profissionais da área. Contudo, pode-se entender Internet das Coisas como uma rede aberta e abrangente de objetos inteligentes capaz de se auto organizar, compartilhar informações, dados e recursos, reagindo e agindo diante de situações e mudanças no ambiente (MADAKAM *et al.*, 2015).

Essa rede de objetos inteligentes que constitui a Internet das Coisas se apoia sobre quatro pilares principais: coisas, pessoas, processos e dados (ZHAMANOV *et al.*, 2017) e, enquanto uma rede global, ela permite a comunicação entre humano-humano, humano-coisas e coisas-coisas. Assim, a IoT se caracteriza como uma revolução tecnológica que representa o futuro da computação e das comunicações (MADAKAM *et al.*, 2015) .

A Internet das Coisas também pode ser vista como uma estrutura na qual todas as coisas têm uma representação e uma presença definida na Internet. De forma mais específica, é notável que a IoT busca em seu uso oferecer novos aplicativos e serviços que possam ser capazes de unir os mundos físico e virtual (MOUHA, 2021), tornando-os ainda mais interconectados.

Nesse contexto, a "coisa" em Internet das Coisas pode ser qualquer dispositivo com qualquer tipo de sensor incorporado com a capacidade de coletar dados e transmiti-los pela rede sem intervenção manual. A tecnologia embutida no objeto ajuda a interagir com os estados internos e o ambiente externo, que por sua vez auxilia no processo de tomada de decisão (MOUHA, 2021).

Assim, dispositivos integrados à Internet das Coisas apresentam características específicas (SASTRA; WIHARTA, 2016) (PALMA *et al.*, 2014), como:

- **Coleta e transmissão de dados:** os dispositivos estão localizados em um ambiente a ser monitorado onde podem coletar e enviar dados para a internet ou para outros dispositivos.
- **Operação baseada em ação:** os dispositivos são programáveis para agir de acordo com qualquer condição.
- **Receber informações:** os dispositivos recebem informações da rede.
- **Comunicação de suporte:** há uma coleção de dispositivos que podem se comunicar entre si por meio de outros nós na mesma rede

Dessa forma, uma grande característica e vantagem da Internet das Coisas é justamente a conexão de diversos dispositivos à internet, sobretudo os da vida cotidiana, como computadores, máquinas de lavar, etc. Isso nos permite acessar informações sobre esses equipamentos em tempo real e de qualquer lugar, facilitando o uso e gerenciamento de diversos aspectos da vida diária (PALMA *et al.*, 2014).

Assim, com o avanço da IoT é possível ter aparelhos ainda mais autônomos, como, por exemplo, os equipamentos multissensoriais que medem temperatura, poluição do ar e umidade (ZHAMANOV *et al.*, 2017). Nesse sentido, outro exemplo interessante é o sensoriamento remoto para controle de incêndio, que usa uma rede de sensores distribuídos por toda a floresta para monitorar a temperatura e evitar possíveis incêndios (PALMA *et al.*, 2014).

Nesse contexto de mudanças e transformações promovidas pela Internet das Coisas, nota-se que as tecnologias que se desenvolvem com base na IoT passam a ser aplicadas também ao cenário de cidades e universidades (BANDEIRA; NETO, 2022), visando uma melhora da

qualidade de vida atual e futura nestes espaços. Surge, assim, o que podemos chamar de *Smart City* e *Smart Campus*.

2.1.1 Smart City

Um projeto de Cidade Inteligente, ou *Smart City*, tem como gênese "a utilização de tecnologias de informação e comunicação para promover a competitividade econômica, a sustentabilidade ambiental e a qualidade de vida dos cidadãos" (INTELI, 2012), centrando-se em áreas como segurança, energia, mobilidade, edifícios, gestão da água e resíduos.

As Cidades Inteligentes geralmente têm como base o uso de Tecnologias da Informação e Comunicação (TICs) para prover serviços essenciais à população e assegurar um futuro mais sustentável. Porém, isso não se limita apenas ao simples uso da tecnologia, havendo uma "visão sistêmica e holística que inclui a sociedade no processo de tomada de decisão e promove a integração de diversos setores, criando um novo modelo de gestão e definição de políticas públicas" (NEVES *et al.*, 2017).

Ainda nesse contexto, alguns autores já sugerem uma evolução do conceito de Cidade Inteligente, que seriam as Cidades Inteligentes e Humanas: *Human Smart Cities* (HSC). Assim, este modelo de cidade mais inteligente e humana visa não apenas o uso da tecnologia em si, mas concentra-se também em ter a população como o maior beneficiado pelo paradigma, prezando pelo bem-estar e desenvolvimento dos cidadãos (NEVES *et al.*, 2017).

Em um modelo de Cidade Inteligente também é imprescindível haver uma análise e integração de dados e informações, dando antecipação e suporte à resolução rápida e eficaz de problemas. Isso ajuda a minimizar os impactos negativos sobre as cidades em diferentes áreas, como segurança pública, gestão de tráfego, serviços de saúde, entre outros (INTELI, 2012).

Dessa forma, "quanto mais informações e conhecimento incorporados ao processo, mais inteligente será a cidade" (JACOSKI; HOFFMEISTER, 2019), sendo esse também um ponto de atenção, já que é desafiador realizar um aproveitamento adequado do alto volume de dados e informações gerados no processo de desenvolvimento e administração de uma *Smart City*.

No âmbito de uma Cidade Inteligente, há possibilidades interessantes como, por exemplo, controle de tráfego em tempo real, gestão inteligente de estacionamento, sistemas para monitoramento do consumo de energia, iluminação inteligente e videovigilância por via remota (INTELI, 2012). Na prática, já é possível ver muitas dessas possibilidades se tornando realidade.

Em um contexto global, o projeto *SmartSantander*[1] é considerado um dos maiores e mais conhecidos testes experimentais de cidades inteligentes do mundo, tendo sido implantado na cidade de Santander, localizada no norte da Espanha. Estima-se que a cidade conta com uma população de aproximadamente 180 mil pessoas.

Para o projeto, Santander foi transformada em um laboratório experimental vivo que agora é usado como uma instalação de teste experimental para pesquisa e experimentação de arquiteturas, tecnologias facilitadoras essenciais, serviços e aplicativos para a Internet das Coisas no contexto das cidades inteligentes (CHENG *et al.*, 2015).

Ao todo, o *SmartSantander* conta com mais de 10 mil sensores distribuídos em toda a infraestrutura da cidade. Alguns dos dispositivos foram anexados a postes de iluminação pública e edifícios, outros estão enterrados no pavimento, como é o caso dos sensores de estacionamento, e há dispositivos instalados também na rede de transporte público da cidade, anexados a ônibus, táxis e carros de polícia.

Os milhares de sensores implantados pelo projeto *SmartSantander* fornecem informações em tempo real sobre diferentes aspectos ambientais da cidade, como luz, temperatura, ruído e CO2, além de outros parâmetros, como a ocupação de vagas em estacionamentos de alguns centros espalhados pela cidade espanhola (CHENG *et al.*, 2015).

Ainda em um contexto global, apenas três cidades brasileiras aparecem no *Smart City Index* (SCI) 2023, ranking realizado pelo *International Institute for Management Development* (IMD). Liderada por Zurich - Suíça, a lista ranqueia 141 cidades, com Brasília, São Paulo e Rio de Janeiro ocupando, respectivamente, as posições 128, 130 e 136 (IMD, 2023). No ranking, o IMD avaliou estruturas e tecnologias empregadas em setores como governança, saúde e mobilidade.

2.1.2 Smart Campus

O Campus Inteligente, ou *Smart Campus*, é baseado no conceito de *Smart City* e tem em sua construção o uso da tecnologia de Internet das Coisas (YANG *et al.*, 2018). Assim, um campus pode ser visto, de fato, como uma pequena cidade independente em vários aspectos. Além disso, universidades e cidades têm questões comparáveis, como problemas de gestão, organização, mobilidade e infraestrutura (PAGLIARO *et al.*, 2016).

Dessa forma, apesar de terem dimensões diferentes, é perceptível que as cidades

[1] https://smartsantander.eu/

e universidades têm estruturas organizacionais, desafios e propósitos similares. Logo, “essa semelhança de perfil, de desafios e de oportunidades de desenvolvimento pode ser um catalisador de soluções compartilhadas ou de desenvolvimento de trabalho colaborativo inter ou intra-instituições” (BANDEIRA; NETO, 2022).

Ainda sobre a proximidade entre *Smart City* e *Smart Campus*, observa-se que a relação do conceito de cidade inteligente aplicado ao campus tem como foco as tecnologias que podem ser utilizadas nas universidades. Assim, as instituições de ensino superior podem ser vistas também como cidades e, por isso, possuem grandes similaridades com as cidades convencionais, principalmente em relação aos desafios enfrentados. Portanto, é por isso que há o interesse em também aplicar soluções inteligentes aos campi universitários (BANDEIRA; NETO, 2022).

Assim, um Campus Inteligente busca integrar um conjunto de tecnologias de inteligência avançada para melhorar a qualidade de vida na universidade através do provisionamento de serviços tecnológicos que são valiosos, dinâmicos e orientados ao usuário para suportar automação e comunicação em tempo real, cobrindo um amplo espectro que inclui: aprendizado, interação social, meio ambiente, economia de energia, etc (MUHAMAD *et al.*, 2017).

Com o uso integrado de ferramentas como Internet das Coisas, *Big Data* e Inteligência Artificial, será cada vez mais possível controlar de forma automatizada aspectos como acessos a espaços, análises de comportamento e adequação de climatização, iluminação e sonorização, além de outras formas de acompanhamento (JACOSKI; HOFFMEISTER, 2019).

Ainda no contexto do *Smart Campus*, alguns autores identificam também subdomínios nesse modelo de Campus Inteligente (NAGOWAH *et al.*, 2019), tais como:

- ***Smart Learning*** **ou Aprendizagem Inteligente:** neste subdomínio as tecnologias são incorporadas para melhorar a experiência de ensino e aprendizagem.
- ***Smart Management*** **ou Gerenciamento Inteligente:** diz respeito às capacidades de gestão de campus, faculdades, departamentos e pessoas.
- ***Smart Buildings*** **ou Edifícios Inteligentes:** as tecnologias são incorporadas aos edifícios como sensores que ajudam a fornecer ambientes internos confortáveis e reduzir custos. Frequentemente este subdomínio é englobado no Gerenciamento Inteligente.
- ***Smart Governance*** **ou Governança Inteligente:** subdomínio que incorpora políticas que regem um Campus Inteligente para garantir eficiência e eficácia

contínuas no ambiente geral do campus.

- ***Smart Classroom* ou Sala de Aula Inteligente:** diz respeito a diferentes condições em uma sala de aula, como temperatura ambiente, luzes acesas ou apagadas, número de alunos presentes, etc.
- ***Smart Health* ou Saúde Inteligente:** representa o estado de saúde geral de diferentes partes interessadas em um campus e inclui técnicas como como telemedicina e bioinformática.
- ***Smart Parking* ou Estacionamento Inteligente:** subdomínio relacionado à disponibilidade e indisponibilidade de vagas de estacionamento e diferentes atividades que ocorrem em uma vaga de estacionamento em um campus.

Nesse contexto, quando comparado a um campus tradicional, o Campus Inteligente apresenta vantagens em diversos aspectos, como: economia de custos e tempo, manutenção automatizada, monitoramento e proteção do meio ambiente, otimização de estacionamento, automação de registro de frequência e detecção de presença e ocupação (ABUARQOUB *et al.*, 2017).

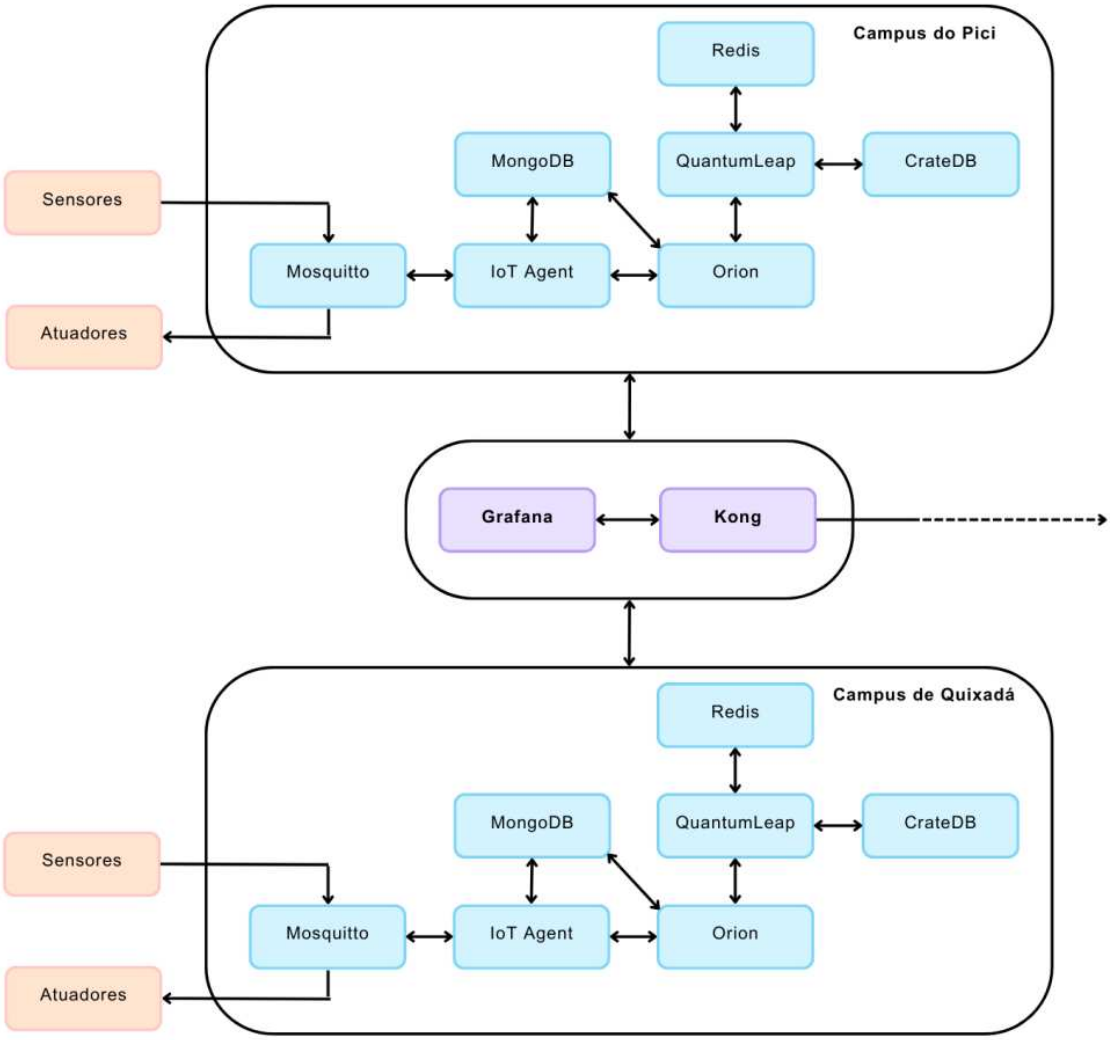
No entanto, observa-se também que há alguns desafios para o Campus Inteligente. É necessária, por exemplo, uma numerosa quantidade de sensores e outros dispositivos IoT para cobrir o campus e, com isso, há também uma grande variedade de captações e dados obtidos por cada um. Além disso, o processamento em tempo desses dados demanda links de comunicação de alta velocidade e poderosas unidades de processamento (ABUARQOUB *et al.*, 2017).

2.2 Campus inteligente da Universidade Federal do Ceará

Com base no que foi visto e discutido anteriormente, o ponto de análise central deste trabalho é o Campus Inteligente da Universidade Federal do Ceará (UFC). Esse projeto de *Smart Campus* foi construído integrando infraestrutura, dados e pesquisadores dos campus do Pici, localizado em Fortaleza, e de Quixadá, no Sertão Central, com planos de expansão para os campi remanescentes.

A Figura 2 apresenta uma visão geral da estrutura atual do *Smart Campus* da UFC:

Figura 2 – Arquitetura do *Smart Campus* da UFC



Fonte: Elaborada pelo autor

A arquitetura do *Smart Campus* da UFC já havia sido construída antes da elaboração deste trabalho e é estruturada da seguinte forma:

Cada campus possui seu próprio *middleware* FIWARE encapsulando sua estrutura e cada instância FIWARE possui uma série de sensores e atuadores:

- Sensores de Temperatura e Umidade
- Sensores de consumo de corrente elétrica
- Atuadores associados a iluminação
- Atuadores associados a projetores
- Atuadores associados a aparelhos de ar-condicionado

Ambas as instâncias FIWARE estão associadas a uma aplicação central que realiza o gerenciamento de tráfego entre as instâncias e integra os dados de ambas. Esse gerenciamento

de tráfego é feito através de uma instância do *Gateway* de API Kong.

2.2.1 Middleware FIWARE

FIWARE[2] é um *middleware* cujo intuito é atuar como uma espécie de abstração para facilitar a integração de múltiplos e diferentes dispositivos e protocolos de comunicação. Sua finalidade é facilitar a existência de um núcleo integrado de aplicações e dados relativos a uma aplicação de IoT. Assim, é possível criar plataformas que suportam o desenvolvimento de soluções inteligentes de forma mais rápida, simples e barata (FIWARE, 2022).

O FIWARE consegue alcançar isso através dos chamados *Generic Enablers* (GEs), que são espécies de bibliotecas que executam uma série de ações dentro da aplicação, a fim de conectar, traduzir e normalizar diferentes protocolos, aplicações externas, além de sensores e atuadores.

Essa conexão é feita através do protocolo de comunicação *Next Generation Service Interfaces* (NGSI), que atua como um protocolo padrão entre os diferentes GEs. Essa conexão é feita com o Orion, um *Context Broker*, sendo o GE que lida com todo o processo de conexão com todos os outros GEs (FIWARE, 2022).

A princípio, essa introdução de um protocolo a mais como dependência da aplicação pode aparentar ser um complicador. No entanto, essa característica garante que quaisquer GEs que tenham sido desenvolvidos pela comunidade funcionem, por padrão, em uma aplicação FIWARE. Assim, garante-se uma grande variedade de funcionalidades que permitem facilitar o processo de desenvolvimento (FIWARE, 2022).

A aplicação FIWARE do projeto de Campus Inteligente da UFC possui os seguintes GEs:

- *Orion*[3]: Como mencionado posteriormente, esse é o GE que gerencia toda a informação de contexto da aplicação FIWARE, sendo fundamental para o funcionamento da aplicação como um todo. Além disso, parte da aglomeração e organização dos dados coletados pela aplicação passam pelo *Orion*, sendo alguns dos endpoints que são consumidos pela aplicação front-end passados por ele.
- *IoT Agent*[4]: O *IoT Agent* é responsável pela tradução de outros protocolos de comunicação para o NGSI. No projeto, atua como um intermediário entre os

[2] https://fiware.org
[3] https://fiware-orion.readthedocs.io/en/master/
[4] https://github.com/FIWARE/tutorials.IoT-Agent

sensores e atuadores e a aplicação FIWARE. Assim, traduz requisições feitas via os protocolos *MQ Telemetry Transport* (MQTT) e *Ultralight*.

- *QuantumLeap*[5]: O *QuantumLeap* é responsável por organizar, manipular e carregar dados históricos. No caso, é utilizado para o armazenamento dos dados de leituras do estado de sensores e atuadores da aplicação. É através dele que são feitas as requisições de leituras de sensoriamento específicas.
- *Mosquitto*[6]: O *Mosquitto* é um *Broker* MQTT de codigo aberto, atraves do qual a comunicação via o modelo *publisher/subscriber* é feita com os sensores e atuadores presentes na arquitetura.
- *Redis*[7]: O *Redis* é uma banco de dados de chave-valor de código aberto, que possibilita armazenar dados simples para acesso rápido posteriormente, frequentemente utilizado para fazer *caching* de informação
- *MongoDB*[8]: O *MongoDB* é um banco de dados orientado a documentos, muito utilizado para o armazenamento de dados não estruturados ou que não possuem uma estrutura bem definida.
- *CrateDB*[9]: Por fim, o *CrateDB* é um banco de dados distribuído, que permite o armazenamento e analise de dados em massa, sendo um dos seus principais casos de uso o armazenamento de dados em tempo real.

A Figura 3 apresenta os *endpoints* do FIWARE utilizados para a elaboração deste projeto.

[5] https://quantumleap.readthedocs.io/en/latest/
[6] https://mosquitto.org/
[7] https://redis.io/
[8] https://www.mongodb.com/
[9] https://crate.io/

Figura 3 – *Endpoints* do FIWARE utilizados.

Tipo	URL	Código	Resposta

GET	{URL FIWARE}:
	{Porta}/campus{Campus}/orion/
	entities?type={Tipo de Objeto}
		200	[{
			id: "String",
			name: { value: "String" },
			description: { value: "String" },
			category: { value: ["String"] },
			location: {
			value: { coordinates: ["Float", "Float"] }
			},
			totalFloors: {
			type: "String",
			value: "Integer",
			}
			}]

GET	{URL FIWARE}:{Porta}/campus{Campus}
	/devices
		200	{
			count: "Integer",
			devices: [{
			device_id: "String",
			entity_name: "String",
			entity_type: "String",
			static_attributes: [
			{
			name: "String",
			type: "String",
			value: "String"
			}
			]
			}]
			}

GET	{URL FIWARE}:{Porta}/campus{Campus}
	/quantumleap/entities/{ID Sensor}
	/attrs/{Atributo}?{Opções}
		200	{
			attr_name: "String",
			entity_id: "String",
			index: ["DateTime"],
			values: ["Float"]
			}

PATCH	{URL FIWARE}:
	{Porta}/campus{Campus}/orion/
	entities/{ID Atuador}/attrs
		204	{
			{Comando}: {
			type: "String",
			value: "String"
			}
			}

Fonte: Elaborada pelo autor

2.2.2 Aplicação Back-end

Implementada em Python, é a aplicação *Back-end* que lida com o armazenamento e gerenciamento das informações de usuários, câmeras de monitoramento e eventos. Através dela é feita a autenticação e autorização do acesso de usuário à aplicação *Front-end*.

Além disso, a transmissão de vídeo das câmeras de monitoramento é acessada através de um *endpoint* desta aplicação. Assim, é possível manipular e utilizar diferentes modelos de *Machine Learning* (ML) que são aplicados sobre as imagens das câmeras para a coleta de certos eventos detectados pelos modelos, como a detecção de diferentes tipos de veículos nas imagens.

A Figura 4 apresenta os *endpoints* da API Python utilizados para a elaboração deste projeto.

Figura 4 – *Endpoints* da API Python utilizados.

Tipo	URL	Código	Resposta
POST	{URL API Python}/token	200	`{ data: { access_token: "String" } }`
GET	{URL API Python}/v1/cameras	200	`[{ id: "String", descricao: "String", latitude: "Float", longitude: "Float", analitico: "String", compiler: "String" }]`
GET	{URL API Python}/analiticos	200	`{ data: { compiler: "String", analitico: "String" } }`
PUT	{URL API Python}/v1/cameras	200	`[{ id: "String", descricao: "String", latitude: "Float", longitude: "Float", analitico: "String", compiler: "String" }]`
GET	{URL API Python}/video_feed/{ID Câmera}	200	Vídeo Requisitado
GET	{URL API Python}/eventos	200	`{ id: "Id", tipo: "String", lp: "String", id_cam: "Id", data: "String", horario: "String" }`
GET	{URL API Python}/imagem/{ID Imagem}	200	Imagem Requisitada

Fonte: Elaborada pelo autor

2.3 Interface de usuário para um *Smart Campus*

Conforme visto nas seções anteriores, a construção e o desenvolvimento de projetos baseados nos modelos de *Smart City* e *Smart Campus* requerem uma quantidade massiva de dados, bem como uma grande quantidade de instrumentos para realizar sua coleta.

Assim, devido a essa alta quantidade de dados, um projeto de cidade ou campus

inteligente requer também uma atenção especial no modo de exibição dessas informações, que deve ocorrer de forma cautelosa e estruturada. Caso contrário, uma ferramenta que normalmente seria extremamente útil para o usuário final pode vir a ser algo confuso e não confiável.

A fim de evitar essa problemática, há no mercado diferentes arquiteturas de aplicações IoT que possuem recursos já pré-embutidos. O objetivo dessas arquiteturas é possibilitar a criação de soluções rápidas e sem grandes custos de desenvolvimento para o projeto. Nesse sentido, algumas das arquiteturas de aplicação IoT que podemos citar são:

- ***Wirecloud***[10]**:** é uma plataforma que tem como propósito proporcionar uma aplicação web centralizada que possibilite ao usuário ter fácil acesso a *dashboards* e visualizações de dados de alta qualidade. *Wirecloud* faz parte do projeto FIWARE, sendo um GE de processamento dentro do mesmo. Por tanto, ele tem uma fácil integração com toda a estrutura da aplicação, facilitando a integração de dados heterogêneos de modo a possibilitar a rápida criação de visualizações e dashboards eficientes, como ilustra o exemplo da Figura 5.

Figura 5 – Exemplo de interface no *Wirecloud* com a visualização de uma métrica e um mapa

Fonte: (WIRECLOUD, 2019)

- ***Grafana***[11]**:** é um software de código aberto para análise, organização e exibição

[10] https://wirecloud.readthedocs.io/en/stable/
[11] https://grafana.com/grafana/

de dados de múltiplas fontes e formatos, possibilitando ao usuário construir *dashboards* e visualizações rapidamente. O *Grafana* possui integração com o FIWARE, sendo, inclusive, recomendado pela documentação do mesmo para a visualização de dados de série temporal através do *CrateDB*. A Figura 6 apresenta um exemplo de interface com várias visualizações criada no *Grafana*.

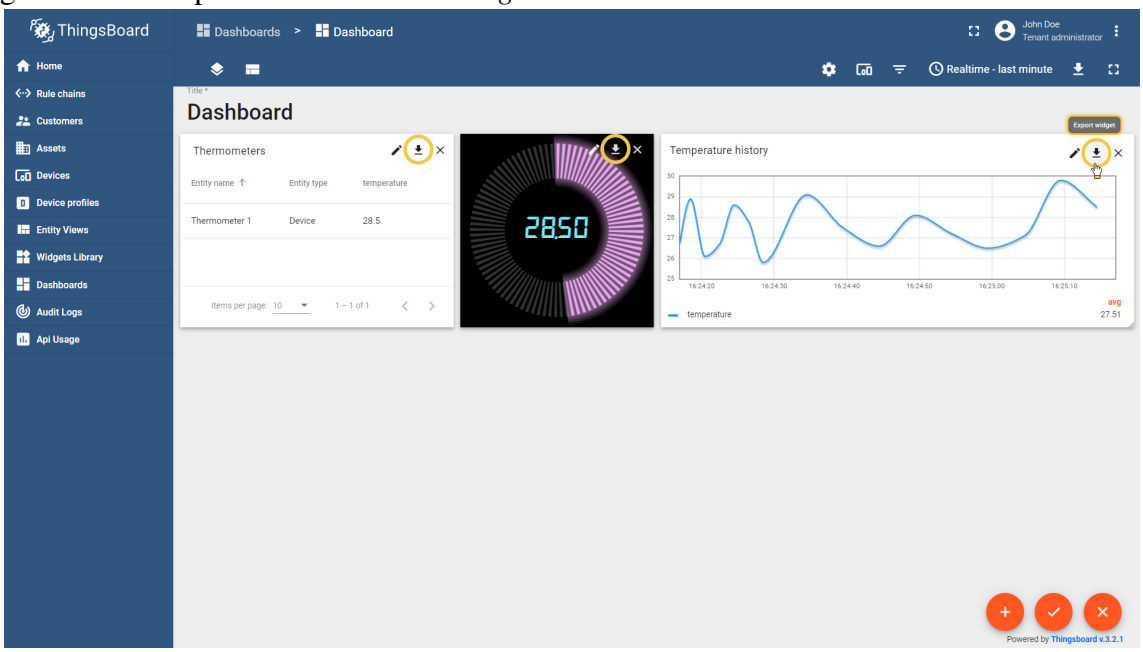
Figura 6 – Exemplo de interface no *Grafana* com várias visualizações

Fonte: (GRAFANA, 2019)

- ***Thingsboard***[12]**:** é uma plataforma IoT de código aberto que disponibiliza gerenciamento de dispositivos, agregação de dados e criação de *dashboards*. Inclui também suporte padrão ao FIWARE e tem um processo simples de integrar com seus dispositivos. A Figura 7 apresenta um exemplo de interface com várias visualizações criada no *Thingsboard*.

[12] https://thingsboard.io/

Figura 7 – Exemplo de interface no *Thingsboard*



Fonte: (THINGSBOARD, 2023)

Apesar dessas soluções simplificarem o processo de elaboração de *User Interfaces* (UIs), há alguns poréns. Por serem generalistas, elas devem ser genéricas o suficiente para se moldarem a qualquer projeto e quaisquer arquiteturas para as quais elas tenham suporte. Assim, essa pode não ser a melhor solução quando comparado a uma aplicação moldada especificamente para um problema em questão.

Dessa forma, para desenvolver soluções customizadas, normalmente é necessário realizar o desenvolvimento de uma aplicação *Front-end* do zero. Com isso, é necessário seguir alguns passos, sendo um dos mais importantes a escolha da linguagem e do *framework* que serão utilizadas para desenvolver a aplicação. Atualmente, essas são algumas das principais opções disponíveis no mercado:

2.3.1 JavaScript

JavaScript[13] é a principal linguagem de programação utilizada no desenvolvimento de aplicações web, permitindo com que uma gama de funcionalidades e interações na web sejam possíveis.

Devido a essa importância, o *JavaScript* foi uma das linguagens que mais cresceram em uso recentemente, sendo a terceira mais usada entre 2022 e maio de 2023, segundo dados do

[13] https://developer.mozilla.org/pt-BR/docs/Web/JavaScript

Índice de Popularidade das Linguagens de Programação, do inglês *PopularitY of Programming Language* (PYPL) (PYPL, 2023).

Com isso, nota-se que o ecossistema *Javascript* tem uma rápida evolução. Afinal, constantemente são desenvolvidas novas bibliotecas e novos *frameworks* com a linguagem, tanto para desenvolvimento *Front-end* quanto para *Back-end*.

Alguns dos principais *frameworks* de *JavaScript*:

2.3.1.1 React.js

React.js[14], também chamado apenas de *React*, é um *framework JavaScript* de código aberto criado pelo *Facebook* para a criação de interfaces de usuário, principalmente as *Single Page Applications* (SPA). É baseado em componentes, o que permite a reutilização rápida e fácil de componentes da interface. Seu principal foco é aumentar a performance através de sua implementação do *Document Object Model* (DOM) virtual.

- Pontos positivos:
  - É fácil de aprender e utilizar;
  - Possui excelente performance devido a sua implementação do DOM virtual;
  - Possui uma comunidade extensa e diversa;
  - Permite uma alta reusabilidade de seus componentes.
- Pontos negativos:
  - Seu ecossistema evolui rapidamente, de forma que várias bibliotecas e conteúdos educativos sobre *React* ficam desatualizados com frequência.
  - Não possui muitas funcionalidades na sua implementação base, sendo comumente necessária a utilização de bibliotecas adicionais.
- Casos de uso:
  - Criação de *Single Page Applications* (SPAs) de larga escala.
  - Criação de componentes de UI altamente reutilizáveis e de fácil manutenção e extensão.
  - Aplicações que requerem de alta performance.

[14] https://react.dev/

2.3.1.2 Angular (2+)

Angular (2+)[15], ou apenas *Angular*, é uma versão completamente reescrita da versão original do *Angular*, e para diferenciar de suas versões mais recentes também é chamado de *Angular.js*. Este é um poderoso *framework JavaScript* desenvolvido pelo *Google* e que utiliza *TypeScript* e uma arquitetura baseada em componentes para construir aplicações escaláveis, manuteníveis e testáveis, sendo um *framework* recomendado para aplicações de pequeno e grande porte.

- Pontos positivos:
  - Sua arquitetura é baseada em componentes que facilitam a reusabilidade.
  - Apresenta uma melhoria de performance considerável se comparado com o *Angular.js* original.
  - Seu ecossistema é grande e possui um forte suporte da comunidade.
  - Possui uma grande gama de funcionalidades e ferramentas sem a necessidade de bibliotecas adicionais.
- Pontos negativos:
  - Tem uma curva de aprendizado maior se comparado com *React* e *Vue*.
  - Requer conhecimento em *TypeScript*.
  - Há alta complexidade de algumas partes do *framework*.
- Casos de uso:
  - Construção de aplicações de grande escala a nível de produção.
  - Construção de aplicações de fácil manutenção e testáveis através de uma estrutura estrita

2.3.1.3 Vue.js

Vue.js[16], também chamado apenas de *Vue*, é um *framework JavaScript* leve utilizado na criação de interfaces de usuários. É facilmente integrável a projetos pré-existentes e, por ser baseado em componentes, também permite a criação de elementos de UI reutilizáveis. O *Vue* combina conceitos de *React* e *Angular*, tornando-se um *framework* versátil.

- Pontos positivos:
  - É leve e flexível.

[15] https://angular.io/
[16] https://vuejs.org/

  - Possui fácil aprendizado e implantação em projetos pré-existentes.
  - Possui uma documentação extensa e completa.
  - O seu ecossistema tem evoluído rapidamente.
- Pontos negativos:
  - Não é tão estável e maduro quando comparado ao *React* e *Angular*.
  - Também possui uma comunidade menor se comparado a ambos.
- Casos de uso:
  - Construção de aplicações pequenas e médias.
  - Tem uma prototipação rápida de componentes de UI.
  - Fácil e rápida integração com projetos pré-existentes.

2.3.2 Escolha da tecnologia do projeto

Dentre as tecnologias citadas acima, o presente trabalho foi desenvolvido em *JavaScript* com *React* pelos seguintes fatores:

- *JavaScript* é nativamente suportado pelos navegadores da web, o que leva a uma melhor performance por si só.
- Devido à virtualização de DOM do *React*, que permite uma maior performance ao lidar com muitas atualizações da UI advindas das múltiplas requisições para carregar os dados dos dispositivos IoT.
- A componentização foi outro fator importante na escolha, já que facilita o reuso e a manutenção do projeto, especialmente caso novos desenvolvedores entrem no projeto, proporcionando uma curva de aprendizagem mais curta e rápida.
- A quantidade de ferramentas e bibliotecas disponíveis na comunidade, fazendo com que a integração do *React* com outras ferramentas e plataformas necessárias para a execução do projeto seja facilmente executada.

3 TRABALHOS RELACIONADOS

Neste capítulo, são discutidos alguns trabalhos relacionados à área em estudo, tanto no desenvolvimento e análise de *Smart Campus* como na elaboração de interfaces e aplicações de *Front-end* para determinados projetos. Para cada trabalho, foi feito um resumo sobre seu propósito e desenvolvimento, fundamentação, bem como um comparativo em relação ao presente trabalho.

3.1 *CampusTalk: IoT Devices and Their Interesting Features on Campus Applications*

O trabalho Lin *et al.* (2018) apresenta a implementação do *CampusTalk*, uma plataforma desenvolvida ao redor da estrutura de um *Smart Campus* que tem o intuito de estimular e difundir o uso e o contato de tecnologias e funcionalidades da Internet das Coisas entre os alunos e outras pessoas dentro do campus.

O *CampusTalk* permite aos alunos desenvolver diversas funcionalidades e aplicações utilizando os dados coletados tanto pelos sensores existentes no campus quanto através dos aparelhos mobile conectados à rede. A plataforma funciona inteiramente pela web, não sendo necessária a instalação de qualquer aplicativo para utilizá-la.

A estrutura do *CampusTalk* permite a criação de aplicações como interfaces customizadas para exibição e análise de diferentes dados que o campus possui, e jogos que utilizam os sensores de aceleração, giroscópio e orientação do dispositivo móvel do usuário para a interação.

O trabalho foca em três aplicações desenvolvidas através do *CampusTalk*:

- *MusicTalk*: Aplicação que reage a diferentes sons ou áudios ao qual o aplicativo mobile do usuário está exposto, exibindo imagens, textos e sons, em resposta. O trabalho menciona o uso dessa aplicação em eventos e comemorações dentro do campus.
- *SmartPhoneTalk*: Aplicação que faz a integração do dispositivo móvel do usuário com a infraestrutura do *CampusTalk*. Através disso, é possível a utilização de sensores do dispositivo móvel para diferentes aplicações, como, por exemplo, interação com um jogo de realidade virtual. Ao acessar é exibido uma interface bem simples que mostra algumas das leituras obtidas pelo dispositivo do usuário.
- *SkeletonTalk*: Aplicação similar ao *SmartPhoneTalk* que permite aos usuários controlar diversos aspectos de uma estrutura geométrica, como formato de hastes

e cores. É possível acessar a aplicação tanto por dispositivos mobile quanto pela web

De forma geral, o trabalho traz uma implementação interessante e criativa de uma aplicação em um *Smart Campus*, tentando aproximar e facilitar o uso dessas tecnologias pelos alunos, algo que também é visado neste presente projeto.

3.2 Expandindo o *SmartCampus*: um guia baseado em Storybook para desenvolvedores

O trabalho Pereira (2021) fala sobre a implementação de uma *dashboard* para visualização de dados de sensoriamento de um *Smart Campus*, executando isso de modo componentizado por meio da ferramenta Storybook. O trabalho faz, então, uma análise sobre o tempo e a facilidade de desenvolvimento e implementação de novas funcionalidades por novos desenvolvedores utilizando alguns desses componentes.

O intuito do trabalho é analisar o impacto que a implementação de elementos de UI de modo componentizado e bem documentado pode ter no desenvolvimento de futuras funcionalidades e na integração de novos desenvolvedores ao projeto. Os dados de experimentação foram levantados por meio da análise da experiência de duas voluntárias que implementaram uma nova página utilizando os componentes e a documentação existente.

O projeto foi desenvolvido utilizando *JavaScript* com o *framework React* devido à sua integração fácil e rápida com o Storybook. Já o presente trabalho, apesar de não utilizar a ferramenta Storybook, também faz a implementação de seus elementos de UI de modo componentizado, a fim de facilitar futuros desenvolvimentos. Para um desenvolvimento futuro, a implementação do Storybook é uma proposta interessante.

3.3 Iniciativa *Smart Campus*: um estudo de caso em progresso na Universidade Federal do Pará

O trabalho Neves *et al.* (2017) apresenta a elaboração do projeto *Smart Campus* UFPA com uma interface móvel chamada Smart UFPA. O trabalho ataca um dos principais problemas levantados pelos estudantes da Universidade Federal do Pará à época: a dificuldade de locomoção e navegação dentro do campus.

Em seu desenvolvimento, o projeto faz o mapeamento dos principais pontos de interesse dentro do campus, além de monitorar a posição e o deslocamento dos ônibus internos

da universidade. Todas essas informações ficam acessíveis por meio da aplicação Smart UFPA.

A aplicação móvel foi desenvolvida para *Android* e faz uso do sistema de mapeamento aberto *OpenStreetMap* (OSM). Isso se deve à valorização e permissão que o OSM dá à comunidade em fazer alterações e adições nos seus dados de mapeamento relativos à gradual adição de pontos de interesse da universidade.

Dentre as futuras melhorias para o trabalho, o autor fala sobre a criação de um *dashboard* de visualizações para os resultados da iniciativa, com base nos dados coletados e implementação de um *middleware* FIWARE. Com isso, o objeto é centralizar e facilitar o desenvolvimento de aplicações e infraestrutura.

Quando comparados, o presente trabalho e o projeto analisado apresentam semelhanças. Ambos os projetos têm em comum o objetivo de sanar um problema similar ao tentar disponibilizar informações do campus universitário de modo mais simples e rápido.

3.4 Uma solução de IoT baseada em FIWARE para gerenciamento de recursos energéticos e serviços acadêmicos em um campus universitário

No trabalho Amurim *et al.* (2021) o autor fala sobre a elaboração e o desenvolvimento da infraestrutura de um *Smart Campus*, parte do que eventualmente viria a ser a atual estrutura do campus inteligente da UFC.

O trabalho propõe uma prova de conceito para a arquitetura do *Smart Campus* da UFC, elaborando uma estrutura de sensores em uma rede *fog*, onde diversos sensores respondem a um mesmo nó computacional que passa o compilado de dados à aplicação central. Além disso, a aplicação web foi implementada com o framework *Vue*.

De modo geral, por tratarem do mesmo projeto de *Smart Campus* da UFC, o presente trabalho e o trabalho analisado apresentam diversos pontos de semelhança, como a estrutura desenvolvida utilizando o FIWARE como *middleware*, além dos sensores, atuadores e câmeras utilizados.

3.5 *Smart Street Light Monitoring and Visualization Platform for Campus Management*

O trabalho Deepaisarn *et al.* (2022) apresenta o desenvolvimento de uma aplicação para monitoramento e visualização da energia e iluminação de um *Smart Campus*.

O projeto monitora várias métricas, mas o trabalho é focado em garantir um uso

melhor e mais eficiente de energia nas *Smart Cities*. Para isso, a implementação no campus universitário é posta como uma etapa de protótipo para, eventualmente, expandir o projeto.

O trabalho foi desenvolvido através de uma aplicação *Node.js* com o *framework* web *Express.js* e os dados colhidos são centralizados e armazenados por uma plataforma externa chamada *CMS Neptune*. Já a aplicação web foi desenvolvida utilizando a biblioteca *Bootstrap*, e as visualizações foram implementadas utilizando a biblioteca *Chart.js* juntamente com o *Leaflet*.

Dentre os próximos passos do projeto está a implementação de um sistema de alertas para casos em que os dados de consumo estejam fora do normal ou quando houver algum problema com o sistema de iluminação. Isso busca garantir a persistência dos dados mesmo em situações onde o sistema acabe ficando fora do ar.

O projeto analisado apresenta alguns pontos de semelhança com o presente trabalho. Entre eles, o uso das bibliotecas *Chart.js* e *Leaflet* a fim de lidar com as visualizações de dados e o mapeamento dentro da aplicação, respectivamente. Outro ponto em comum é o objetivo de disponibilizar uma melhoria sobre o processo de monitoramento e acompanhamento das métricas do *Smart Campus* em estudo.

3.6 *Smart Campus as a living lab on sustainability indicators monitoring*

No trabalho Negreiros *et al.* (2020) é proposto o desenvolvimento da arquitetura de um *Smart Campus* a fim de alcançar os Objetivos de Desenvolvimento Sustentável (ODS) propostos pelas Nações Unidas.

Para tal, o trabalho apresenta a proposta dos campi inteligentes serem uma espécie de protótipo para as cidades inteligentes, por apresentarem desafios similares relativos à gerência, inovação, pesquisa, infraestrutura e mobilidade. Desse modo, trabalhos de pesquisas que tiverem sucesso nos campi poderiam ser eventualmente expandidos e escalados para funcionarem em *Smart Cities*.

O projeto ainda levou a uma parceria entre a universidade FACENS e a empresa OsiSoft com o objetivo de utilizar a aplicação PI Software no projeto de monitoramento do campus.

O intuito do projeto é observar se os dados necessários para estar de acordo com os indicadores do *GreenMetric World University Ranking* estão presentes e se são facilmente acessíveis na aplicação de visualização:

- Número de fontes de energia renováveis no Campus.

- Fontes de energia renovável e suas capacidades - Energia Solar.
- Consumo de energia por ano.

Dessa forma, o trabalho concentra-se na quantidade e nos tipos de dados disponíveis para visualização. Ao contrário do presente trabalho, o projeto analisado não dá foco à formatação e estruturação de como as visualizações de dados são exibidas e acessadas pelo usuário. Logo, não foram encontradas grandes semelhanças entre os dois trabalhos.

3.7 *Front-end* web para uma ferramenta de experimentação em cenários de *Mobile Cloud Computing*

No trabalho Pereira (2022) foi desenvolvido uma aplicação *Front-end* web para a aplicação *MCC Testbed*. Essa aplicação existe para tratar do problema de restrição de poder computacional de aparelhos móveis utilizando uma técnica chamada *offloading*, onde parte ou todo o processamento de uma atividade é passado para a nuvem.

O intuito do trabalho foi desenvolver uma interface mais clara e otimizada, a fim de simplificar e auxiliar no uso da aplicação *MCC Testbed* e da interface original. A aplicação foi implementada e testada pelos desenvolvedores do *MCC Testbed* e, após ser aprovada, foi disponibilizada para uso geral.

A aplicação foi desenvolvida em *JavaScript* no *framework React*, fazendo uso das bibliotecas do *Material UI* e *Styled Components* para o desenvolvimento do design da aplicação. Além disso, foi utilizada a biblioteca *Axios* para as requisições entre a aplicação e a API do *MCC Testbed* e do *Electron* para o desenvolvimento de uma aplicação *desktop*.

Quando comparados, o presente trabalho e o projeto analisado possuem semelhanças quanto ao objetivo de desenvolver uma aplicação web que disponibilize de forma simples e acessível ao usuário final uma interação com uma certa aplicação ou volume de dados.

3.8 Sistema de monitoramento para área verde de um campus universitário inteligente

O trabalho Feitosa (2022) apresenta o desenvolvimento de um projeto para o monitoramento das áreas verdes de um campus universitário inteligente, com foco nos seguintes dados:

- Temperatura.
- Umidade.

- Pluviosidade.

O intuito do projeto, além de desenvolver um sistema que monitore esses dados, é desenvolver uma prova de conceito utilizando o FIWARE como *middleware* para a organização e o desenvolvimento a longo prazo.

O trabalho justifica sua implementação devido à quantidade e variedade de diferentes ferramentas que são utilizados nos contextos de IoT e *Smart Campus*, de modo que sem a utilização de uma ferramenta como o FIWARE o desenvolvimento se torna cada vez mais complexo e lento.

Uma grande semelhança entre o projeto analisado e o presente trabalho é justamente a estrutura montada para o *Smart Campus* em estudo, com a utilização do FIWARE como *middleware* e a organização da aplicação como um todo.

3.9 Comparativo dos trabalhos apresentados

Para apresentar uma visão geral dos temas discutidos até então, a Tabela 1 apresenta um quadro comparativo entre o conjunto de trabalhos vistos neste capítulo e o presente trabalho:

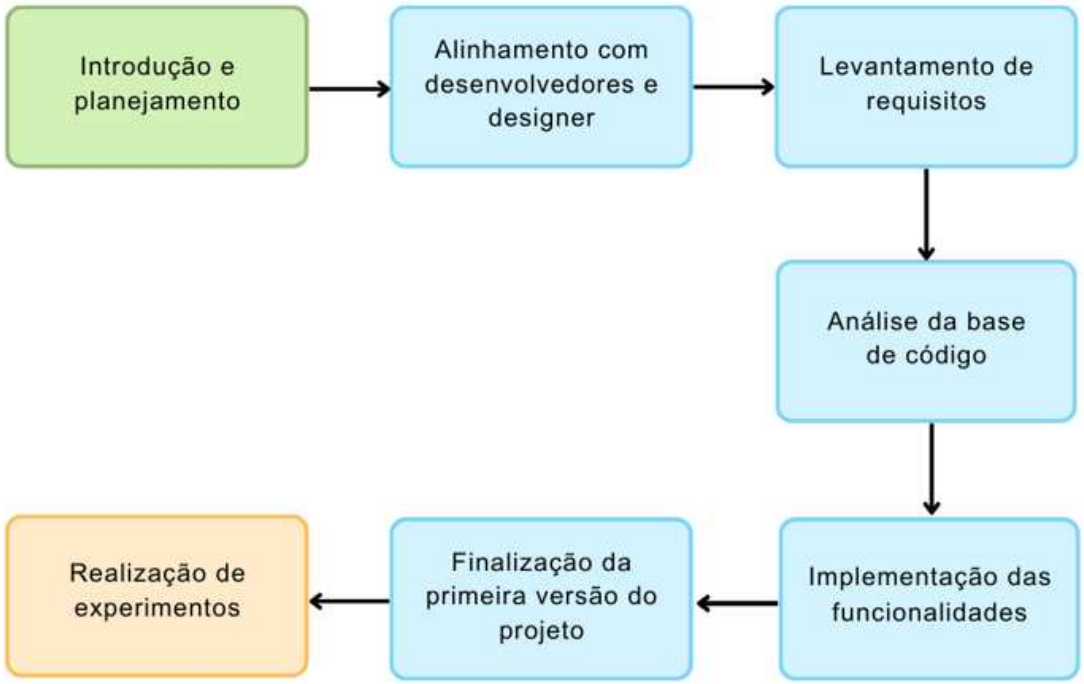
Tabela 1 – Comparativo entre os trabalhos relacionados e o presente trabalho

	Tecnologias Frontend	**Foco em Smart Campus?**	**Tipos de visualizações**	**Possui interação com dispositivos inteligentes**
(LIN *et al.*, 2018)	-	Sim	Por ser uma aplicação educativa, fica a critério do aluno decidir o que pretende visualizar e utilizar	Sim
(PEREIRA, 2021)	React.js, Storybook	Sim	Trata do processo de desenvolvimento, então não possui visualizações específicas.	Não
(NEVES *et al.*, 2017)	OpenStreetMap, Osmdroid	Sim	Mapas	Não
(AMURIM *et al.*, 2021)	Vue.js	Sim	Gráficos de barra e de linha	Sim
(DEEPAISARN *et al.*, 2022)	Bootstrap, Leaflet, Chart.js	Sim	Gráficos de linha, Mapas	Não
(NEGREIROS *et al.*, 2020)	PI System, software proprietario da OsiSoft	Sim	Gráficos de barra, linha, circular, Videos	Não
(PEREIRA, 2022)	React.js, Material UI, Styled Components	Não	Telas voltadas para a utilização do MCC Testbed	Não
(FEITOSA, 2022)	-	Sim	-	Não
Este trabalho	React.js, Material UI, Leaflet, Chart.js	Sim	Gráficos de barra, Mapas, Videos, Fotos	Sim

4 METODOLOGIA

Neste capítulo, são apresentadas as etapas planejadas e executadas para por em prática este trabalho e atingir os objetivos definidos. Assim, cada seção do capítulo analisa uma etapa da metodologia adotada no desenvolvimento do presente projeto. A Figura 8 ilustra, por meio de um fluxograma, as atividades desenvolvidas na construção do trabalho.

Figura 8 – Sequenciamento da metodologia utilizada na execução deste projeto.



Fonte: Elaborada pelo autor

4.1 Introdução e planejamento

Como mencionado anteriormente, boa parte da infraestrutura do projeto de *Smart Campus* da UFC já havia sido desenvolvida antes mesmo deste trabalho ser iniciado. Além disso, previamente já havia sido realizada uma prova de conceito da interface utilizada neste trabalho.

Nesta prova de conceito havia sido implementada uma versão inicial da tela de mapeamento, com apenas a exibição do mapa e o carregamento parcial das informações das câmeras de uma das APIs *back-end*.

Dessa forma, foi com base nesse protótipo que o presente trabalho foi construído e teve início. Logo, em um primeiro momento, foi necessária uma introdução ao projeto para entender em que contexto ele estava inserido, bem como havia sido seu desenvolvimento até

então.

Para isso, foram realizadas algumas reuniões com o gerente de projeto para fins de introdução e planejamento. Desta forma, foi possível identificar para este trabalho quais as problemáticas e os desafios enfrentados pelo projeto, assim como possíveis soluções.

Ainda nesta etapa, foi inicada uma pesquisa e análise de outros projetos já publicados acerca de *Smart Campus* e *dashboards* relacionadas a esse tema. O objetivo era encontrar boas referências que pudessem guiar o desenvolvimento deste trabalho.

4.2 Alinhamento com desenvolvedores e designer

No início desta etapa já era possível ter um bom entendimento sobre o contexto do projeto, suas problemáticas, desafios e possíveis soluções, além de uma boa base referencial sobre outros projetos de *Smart Campus*.

Dessa forma, na etapa atual foram realizadas reuniões de alinhamento com os colaboradores envolvidos com o projeto do campus inteligente da UFC até então, sendo eles dois desenvolvedores e uma designer.

Com o projeto de *Smart Campus* já bem desenvolvido e com o protótipo de interface pré-existente, haviam muitas informações que precisavam ser compreendidas junto aos desenvolvedores antes de dar continuidade ao trabalho. Entre essas informações, pode-se citar:

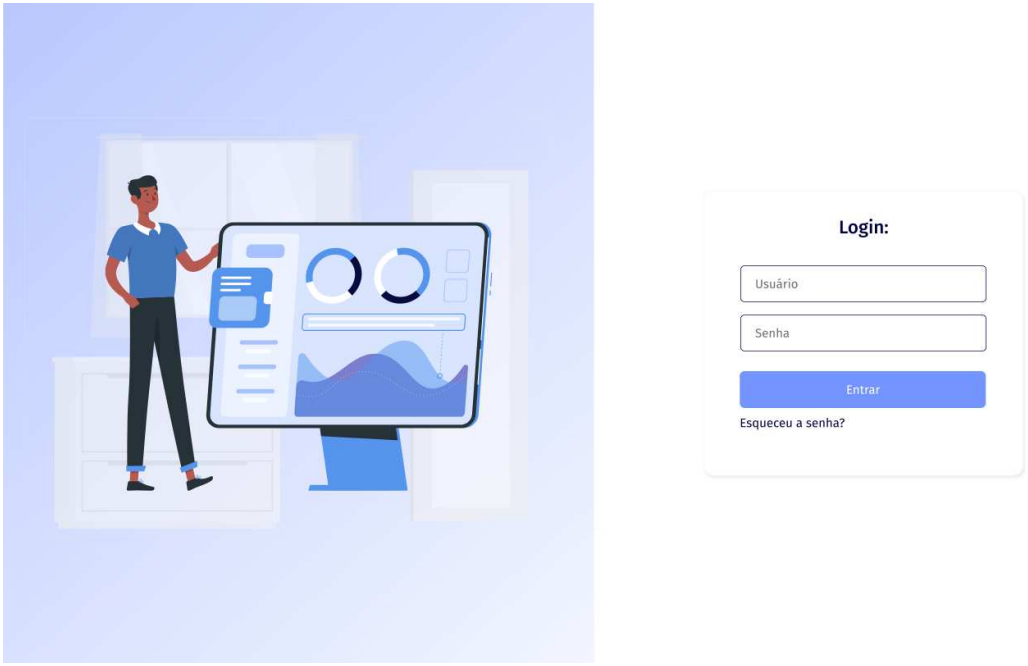
- Tipos de dados monitorados
- Funcionalidades implementadas no protótipo de interface web
- Quais as tecnologias envolvidas na implementação do *Smart Campus* e do protótipo

Já no alinhamento com a designer, foi possível conhecer e entender melhor a proposta de design desenvolvida para a interface web do projeto. Isso foi fundamental para o desenvolvimento deste trabalho, pois o design proposto foi seguido em boa parte da elaboração do projeto.

No entanto, como o design proposto para a interface web é extenso e completo, ele já apresenta, inclusive, funcionalidades que ainda não foram implementadas nesta primeira versão do projeto. Assim, o design proposto é importante para guiar também desenvolvimentos futuros da aplicação.

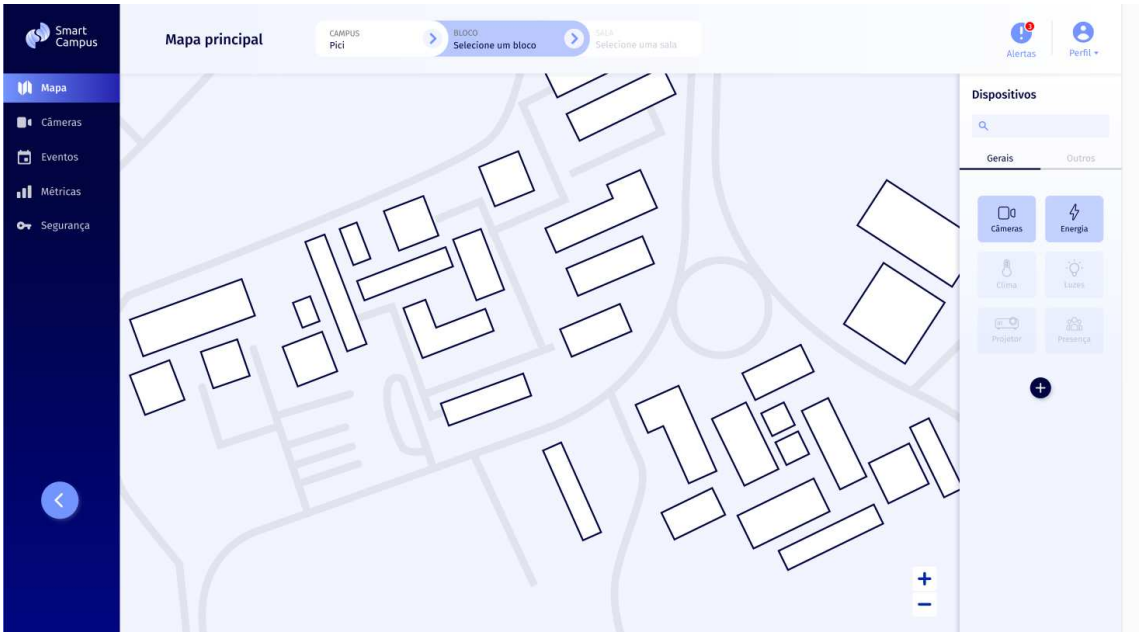
As Figuras 9, 10 e 11 destacam algumas partes da proposta de design para a interface web do projeto:

Figura 9 – Design da tela de login da aplicação web.



Fonte: Elaborada pelo autor

Figura 10 – Design da tela inicial após login.



Fonte: Elaborada pelo autor

Figura 11 – Design da exibição da imagem de uma câmera no mapa.

Fonte: Elaborada pelo autor

4.3 Levantamento de requisitos

Com um melhor entendimento do estado do projeto, esta etapa foi dedicada à realização de uma entrevista com alguns dos principais clientes e utilizadores da interface, que, neste primeiro momento, seriam outros professores e coordenadores da UFC.

Assim, foi feita uma série de perguntas a esses clientes e usuários, abordando aspectos como expectativas, funcionalidades e elementos da UI do projeto:

1. Quais os três principais dispositivos inteligentes e suas métricas que considera mais importantes, em ordem, no monitoramento de uma sala de aula?
2. Quais as visualizações mínimas de cada dispositivo inteligente que já seriam suficientes para seu uso?
3. Quão customizável ou modificável gostaria que essas visualizações fossem?
4. Quão interessante seria poder atuar sobre certos dispositivos através da aplicação?
5. Gostaria de receber notificações para alguns tipos de eventos relacionados às medições? Se sim, quais?
6. Na sua opinião, quais seriam as outras pessoas que, além de você, teriam interesse em utilizar essa aplicação?
7. Na sua opinião, faz sentido usuários terem acesso a dados de outros campi, blocos, salas além do de seu contexto?

8. Quais tipos de usuário fazem sentido existir na aplicação?
9. Em qual plataforma você faria mais uso da aplicação: dispositivos móveis ou web?
10. Na sua visão, que tipo de ganho haveria com o uso dessa aplicação?

Com base nas perguntas realizadas aos entrevistados, foi possível realizar o levantamento de requisitos para o desenvolvimento da interface. Esses requisitos podem ser ordenados da seguinte forma:

1. O usuário deve conseguir ter acesso aos dados de quaisquer campi, bloco ou sala, pelo menos na primeira versão.
2. A tela inicial da aplicação, após o usuário passar pelo processo de autenticação, deve apresentar uma visualização com um mapa centralizado no campus atual.
3. Os dispositivos mínimos para a versão inicial devem ser:
    - Temperatura
    - Umidade
    - Câmeras
    - Luzes
4. As visualizações devem cobrir os seguintes dados:
    - Medições nas últimas 24 horas
    - Medições nos últimos 31 dias
    - A última medição feita
    - A média de medições do dia atual
5. As visualizações não serão interativas, pelo menos na primeira versão
6. Os usuários devem ser capazes de alterar o estado de um atuador disposto no mapa.
7. Na primeira versão não é necessário priorizar um sistema de alertas e notificações. Contudo, isso deve ser prioridade nas próximas versões.
8. Na versão inicial haverá apenas um tipo de usuário comum, mas em versões posteriores deve haver, ao menos, usuários base e usuários administradores.

4.4 Análise do código fonte

Nas etapas anteriores foi possível contextualizar o projeto e elencar os principais requisitos para a sua primeira versão. Agora, nesta etapa atual, foi realizada uma análise da base

de código que já havia sido construída previamente por outros desenvolvedores do projeto, a fim de compreender o contexto atual do protótipo, sua implementação e funcionalidades.

Dessa forma, para realizar a análise da base de código do protótipo pré-existente, foram executados os seguintes passos:

1. Verificar se o projeto, na versão atual, estava funcional. Caso não, realizar a correção dos problemas;
2. Observar e estudar as funcionalidades implementadas, verificando o que se encaixa ou não com os requisitos levantados na etapa anterior.
3. Analisar as tecnologias utilizadas, removendo ou adicionando tecnologias conforme necessário.
4. Verificar o código de forma geral, analisando se existem grandes refatorações a serem feitas nas funcionalidades presentes. Este passo implica na qualidade de código em si, não em alteração de funcionalidades.

Com esta etapa, foram realizadas uma serie de ações a fim de ajustar o código fonte para facilitar o desenvolvimento das novas funcionalidades. Dentre as alterações que foram feitas estão:

- **Extração de código duplicado para seus respectivos componentes:** Essa extração foi executada para as implementações da barra de navegação e da barra de usuários. Elas foram extraídas para os componentes chamados *Sidebar* e *Userbar*, respectivamente.
- **Remoção do uso de *TypeScript* em favor do *JavaScript*:** Essa escolha foi feita por preferencia pessoal e pela falta de familiaridade com o uso do *TypeScript*, a fim de evitar problemas com a sintaxe específica do *TypeScript*, podendo assim, iniciar o desenvolvimento de funcionalidades mais rapidamente.
- **Adição de bibliotecas de análise de código estático:** Adição do *ESLint*[1] e *Prettier*[2] ao projeto, duas bibliotecas de análise de código estático, de modo a corrigir inconsistências no código existente, além de garantir a padronização e consistência do código fonte ao longo do desenvolvimento.
- **Alteração no manuseio de rotas na aplicação:** Foi criado um componente *Dashboard* como uma interface padronizada nas rotas de aplicação para garantir e controlar a renderização de certos componentes visuais sem a necessidade de

[1] https://eslint.org/
[2] https://prettier.io/

adicionar chamadas explícitas a esses componentes. Os componentes utilizados no *Dashboard* foram *Sidebar*, *Userbar* e *Propertiesbar*.

- **Adição das bibliotecas *Material UI*[3] e *Material Icons*[4]:** Essa escolha também foi feita por preferencia pessoal e por familiaridade com o uso dessas bibliotecas, sendo possível utilizar seus componentes para agilizar o desenvolvimento das funcionalidades do projeto.

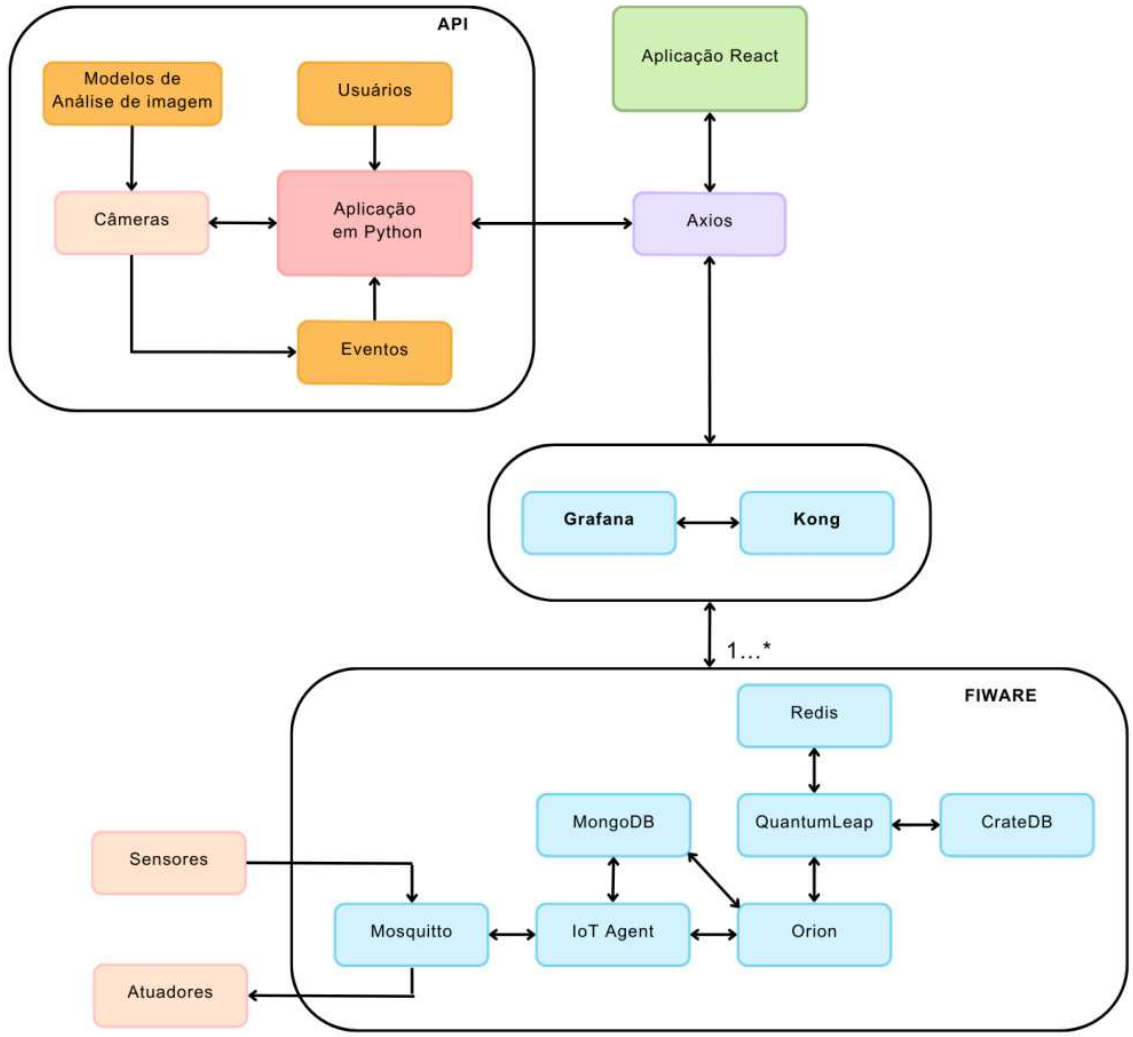
Com a realização dos ajustes acima, a partir deste ponto já é possível iniciar a implementação, no projeto-base, das funcionalidades relacionadas aos requisitos levantados.

4.5 Implementação das funcionalidades

Nesta etapa, teve início o processo de implementação das principais funcionalidades da primeira versão da interface web para campus inteligente desenvolvida neste trabalho. A implementação teve como base o levantamento de requisitos realizado na etapa anterior.

[3] https://mui.com/
[4] https://mui.com/material-ui/material-icons/

Figura 12 – Diagrama geral das aplicações envolvidas no projeto.



Fonte: Elaborada pelo autor

Conforme ilustrado no diagrama da Figura 12, a aplicação se comunica com duas outras aplicações para requisitar os dados necessários à apresentação das telas. Os dados relativos aos usuários, câmeras e eventos são extraídos da aplicação *Back-end* em *Python* através de uma API REST. Já os dados relativos aos blocos acadêmicos, salas, sensores, atuadores, respectivas medições e estados são extraídos da aplicação FIWARE.

Com esses dados, as seguintes telas e visualizações foram implementadas ao longo desta etapa a fim de entregar todos os requisitos elencados:

- Tela de autenticação;
- Tela de mapeamento;
  - Visualizações dos sensores de temperatura e umidade e suas respectivas métricas;
  - Visualizações dos atuadores de energia e suas respectivas ações;

  - Visualizações das câmeras e suas respectivas imagens e ações;
- Tela de listagem de eventos;
- Tela de listagem de câmeras.

As próximas seções abordam o desenvolvimento de cada uma dessas telas.

4.5.1 Tela de autenticação

A tela de autenticação ilustrada na Figura 13 é composta por um componente *Login* que lida com a requisição de autenticação através da biblioteca *JavaScript Axios* e com a manipulação dos respectivos estados caso a requisição de autenticação tenha sucesso ou não.

Figura 13 – Página de autenticação da aplicação.

Fonte: Elaborada pelo autor

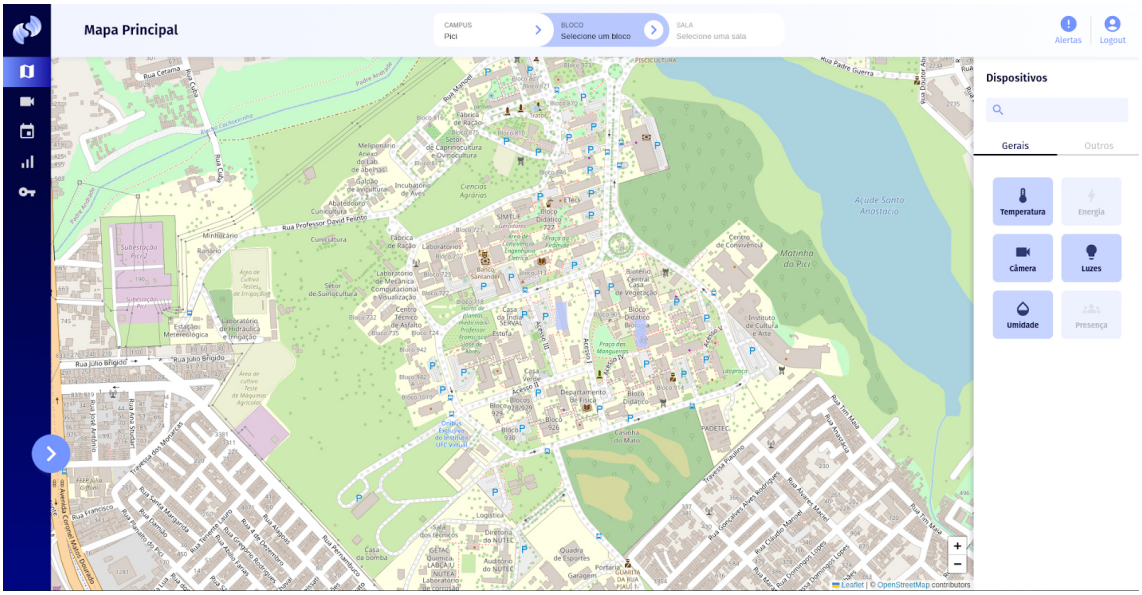
Ao preencher o nome de usuário e a senha, uma requisição *POST* é feita através do *Axios* para a aplicação *Back-end*, onde é verificado se o usuário possui acesso ao restante da aplicação. Caso a aplicação tenha sucesso, o usuário é redirecionado para a tela de mapeamento, caso contrário, o formulário é atualizado com uma mensagem de erro com o motivo da falha.

O componente *Login* presente na tela de autenticação foi desenvolvido utilizando componentes *Button* e *Form* da biblioteca *React Bootstrap*.

4.5.2 Tela de mapeamento

A tela de mapeamento ilustrada na Figura 14 é a principal tela da aplicação. Nela, o usuário tem acesso às informações dos campi do Pici, Quixadá, Sobral e Benfica, com seus respectivos blocos acadêmicos e respectivas salas registrados no FIWARE.

Figura 14 – Página inicial da aplicação exibindo o mapa e seus dispositivos.

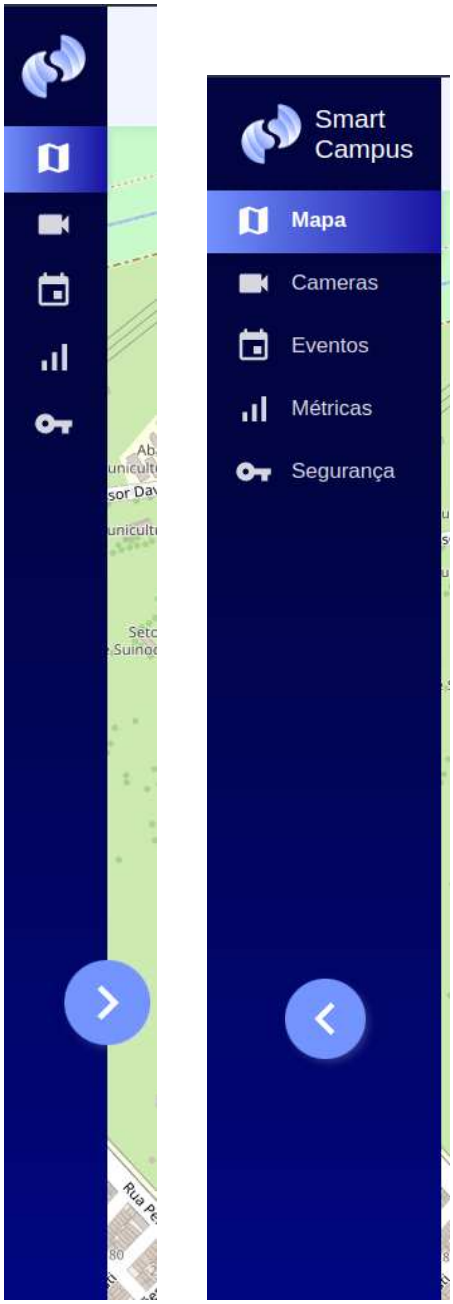


Fonte: Elaborada pelo autor

4.5.2.1 Componente Sidebar

A barra lateral à esquerda da página é o componente *Sidebar*, que está presente em toda a aplicação, exceto na tela de autenticação. É através dele que é feita a navegação pelas diferentes telas. É possível interagir com esse componente aumentando ou diminuindo a barra conforme a necessidade, como ilustra a Figura 15.

Figura 15 – Componente *Sidebar* fechado e aberto.



Fonte: Elaborada pelo autor

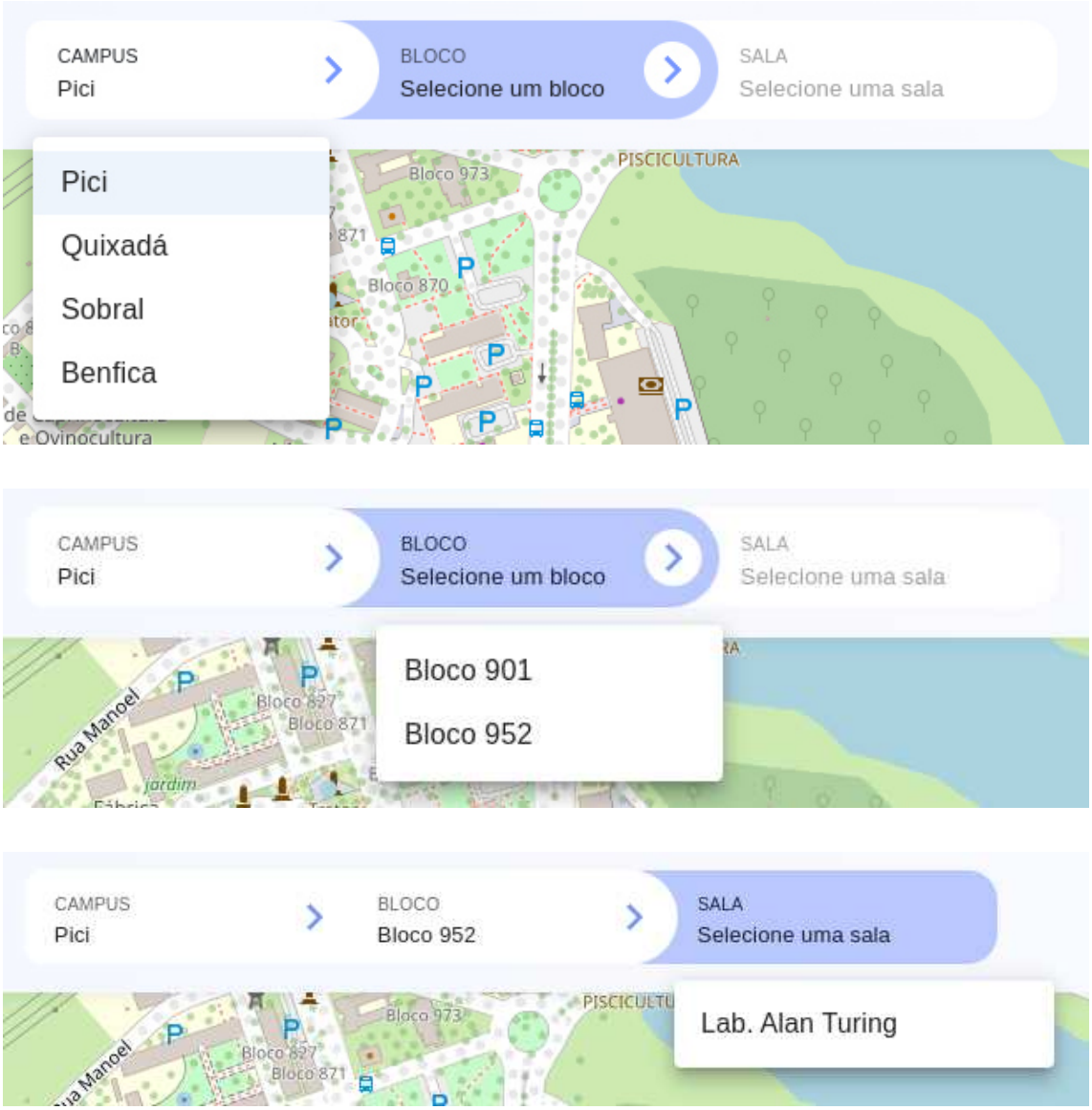
As opções de métricas e segurança da *Sidebar* não foram implementadas nesta versão. Ao clicar nelas, o usuário é redirecionado para a tela de mapeamento. Elas foram adicionadas por fazerem parte do design inicial, mas serão revisadas nos desenvolvimentos futuros da aplicação.

4.5.2.2 Componente Userbar

O segundo componente de navegação está localizado na parte superior da tela: o componente *Userbar*. Por meio dele é possível alterar o campus, sala e bloco atual, nesta

sequência, como ilustra a Figura 16. Esses valores são armazenados no armazenamento local do navegador do usuário, mantendo a escolha do usuário caso a página seja recarregada.

Figura 16 – Componente *Userbar* com seletores de campus, blocos e salas abertos.



Fonte: Elaborada pelo autor

Ao acessar esta tela, uma série de requisições é feita para carregar os dados de contexto necessários para exibir e ter acesso às informações necessárias. Assim, os seguidos dados são carregados:

- Blocos acadêmicos do campus selecionado
- Dispositivos sensores e atuadores do campus selecionado
- Câmeras do campus selecionado

Todos esses dados são armazenados através da API de contexto do *React*, tornando-os

acessíveis a todos os componentes dentro do contexto em que os dados foram carregados.

Esse fator é relevante porque a filtragem de campus, bloco e sala pode ser utilizada em outras páginas, para além da tela de mapeamento. Assim, é necessário haver um acesso rápido e fácil desses dados para o desenvolvimento rápido e eficaz dessas telas.

4.5.2.3 Componente Propertiesbar

O último componente de navegação localizado na tela de mapeamento é o componente *PropertiesBar*, em que os diferentes tipos de dispositivos são listados. Assim, o usuário é capaz de habilitar e desabilitar os tipos de dispositivos a serem exibidos no mapa. A Figura 17 apresenta um exemplo de uso do *PropertiesBar* com a seleção e exibição de dois dispositivos.

Figura 17 – Componente *Propertiesbar* com os dispositivos de Temperatura e Luzes selecionados e sendo exibidos no mapa.

Fonte: Elaborada pelo autor

Todas as interações com os componentes *Userbar* e *PropertiesBar* alteram diretamente os dados exibidos no mapa, alterando, assim, quais tipos de dispositivos devem ser exibidos e a qual campus eles pertencem.

4.5.2.4 Exibição e interação com o Mapa

Ao carregar a tela, inicialmente é renderizado o mapa centralizado sobre o campus do Pici, podendo ser reposicionado, alterando o campus atual no componente *Userbar*, conforme mencionado previamente.

O mapa foi implementado utilizando a biblioteca *Leaflet*[5]. Com ela, é possível exibir

[5] https://leafletjs.com/

marcadores e polígonos utilizando os componentes *Marker* e *Polygon*, respectivamente. Com os dados dos sensores, atuadores e câmeras do campus atual carregados, a exibição desses elementos no mapa é feita através do componente *Marker*, onde cada tipo de dispositivo possui seu próprio ícone e diferentes ações ao clique do usuário. Caso o usuário altere os tipos de positivos selecionados no componente *Propertiesbar*, serão exibidos no mapa apenas os marcadores do dispositivos selecionados.

Da mesma forma, com os dados dos blocos acadêmicos do campus atual carregados, a exibição deles no mapa é feita através do componente *Polygon*. No entanto, atualmente os blocos possuem apenas as coordenadas de origem nos seus dados de localização, sendo feita, então, uma extrapolação com base nesse ponto para fazer um desenho do bloco estimado no mapa. O bloco selecionado no componente *Userbar* possui sua cor alterada no mapa para identificar qual está selecionado.

Um dos problemas encontrados durante o desenvolvimento, foi o fato de que a grande maioria dos dispositivos podem ser localizados muito próximos um do outro, fazendo com que a visualização de um bloco acadêmico possa ficar confusa. Para lidar com isso, foi utilizada a biblioteca *React Leaflet Markercluster*[6], que permite exibir marcadores próximos como apenas um marcador, denotando o agrupamento de dispositivos de determinada região, com esse agrupamento englobando mais ou menos dispositivos conforme o usuário altera a magnificação do mapa ou interage com os agrupamentos.

4.5.2.5 Componente ReadingsModal

No mapa, é possível interagir com os sensores, atuadores e câmeras exibidos. Ao clicar em um dos sensores, o componente *ReadingsModal* é exibido na tela. Esse componente é um modal que exibe as leituras coletadas para os dados do respectivo dispositivo clicado.

Conforme elencado nos requisitos, o modal exibe as seguintes medições:

- Medições das últimas 24 horas
- Medições dos últimos 31 dias
- Última medição feita até então
- A média de medições do dia atual

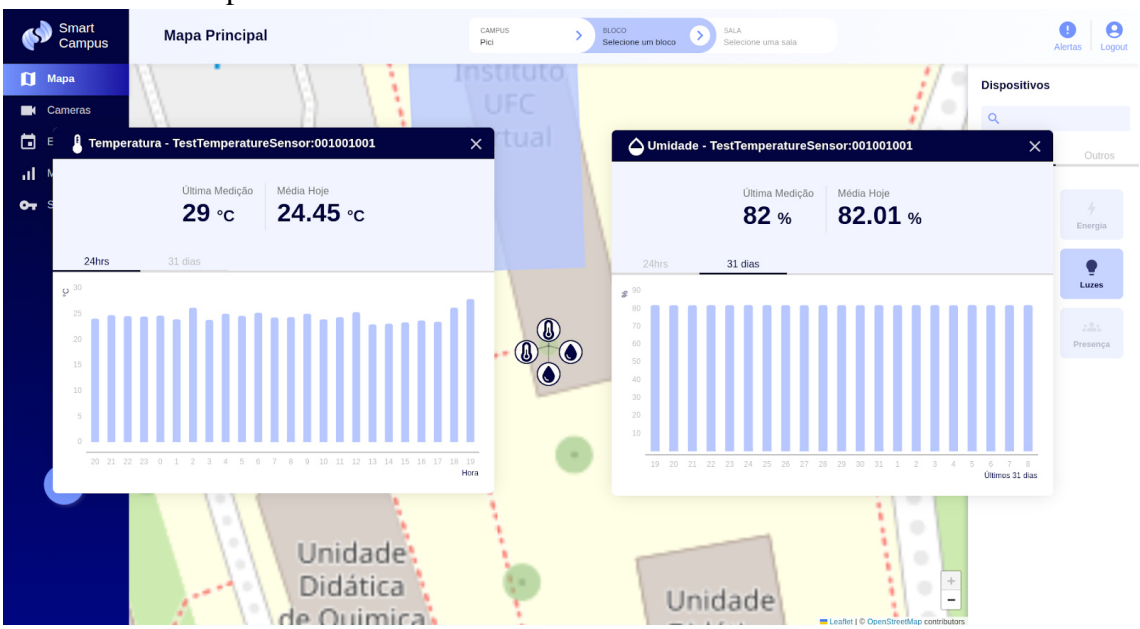
A implementação dessas visualizações é feita através da biblioteca *React Chart.js*

[6] https://github.com/YUzhva/react-leaflet-markercluster

2,[7] que é a implementação da biblioteca *Chart.js*[8] para o *React*.

Além disso, o modal que exibe as medições é totalmente interativo, de modo que o usuário pode arrastá-lo por toda a tela e posicioná-lo do modo que preferir. Essa funcionalidade foi implementada fazendo uso da biblioteca *React Draggable*[9]. Com essa funcionalidade, também é possível exibir múltiplos modais ao mesmo tempo, como mostra a Figura 18.

Figura 18 – Duas instâncias do componente *ReadingsModal* com as respectivas medições de dois dispositivos diferentes.



Fonte: Elaborada pelo autor

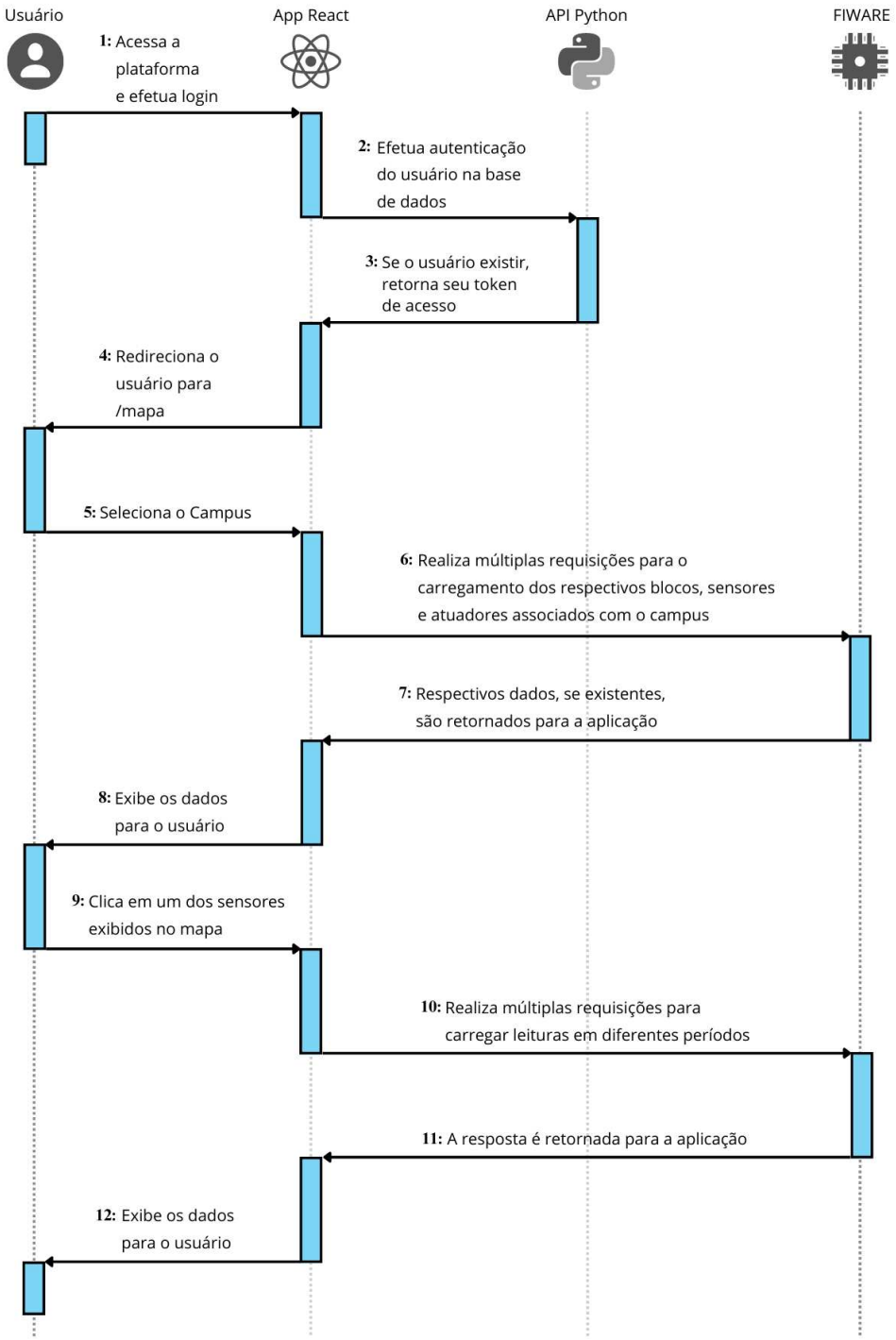
O carregamento dos dados exibidos é feito através de três requisições para a aplicação FIWARE:

- Requisição para buscar as medições das últimas 24 horas agrupadas por hora.
- Requisição para buscar as medições dos últimos 31 dias agrupadas por dia.
- Requisição para buscar a última medição feita até então.

O diagrama da Figura 19 ilustra a sequência de requisições necessárias para a exibição de medições no componente *ReadingsModal*:

[7] https://react-chartjs-2.js.org/
[8] https://www.chartjs.org/
[9] https://github.com/react-grid-layout/react-draggable

Figura 19 – Diagrama de sequência das requisições de exibição de medições no componente *ReadingsModal*



Fonte: Elaborada pelo autor

Assim que esses dados são coletados, o componente passa de um estado de carregamento para o estado de exibição, exibindo os dados para o usuário. Além disso, caso não existam dados dentro dos períodos requisitados, é exibido um estado específico que passa esta informação para o usuário, como mostra a Figura 20.

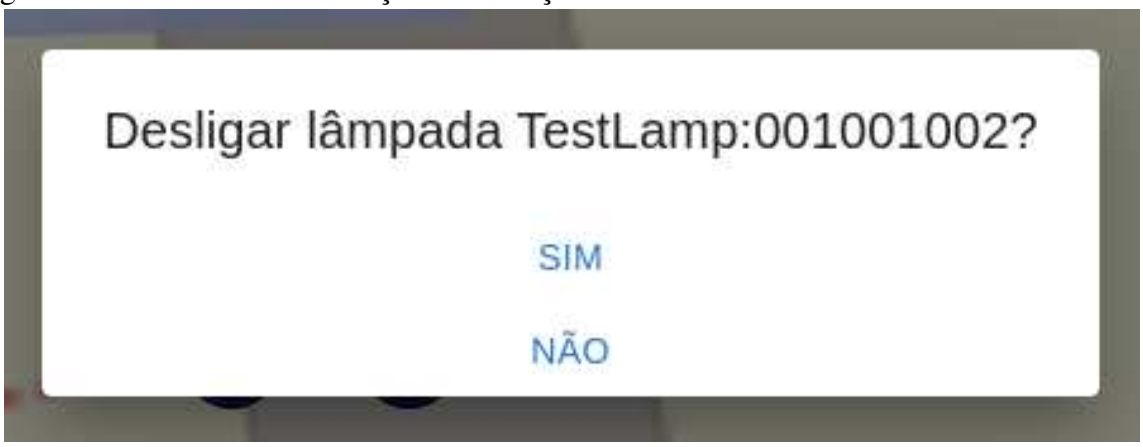
Figura 20 – Componente *ReadingsModal* mostrando um dispositivo que não possui medições para os períodos requisitados

Fonte: Elaborada pelo autor

4.5.2.6 Dispositivos atuadores

Como mostra a Figura 21, ao clicar em um atuador, é exibido um modal de confirmação para o usuário verificar se a alteração de estado que ele pretende realizar está correta. Caso o usuário confirme, a alteração do estado do atuador é exibida diretamente no mapa.

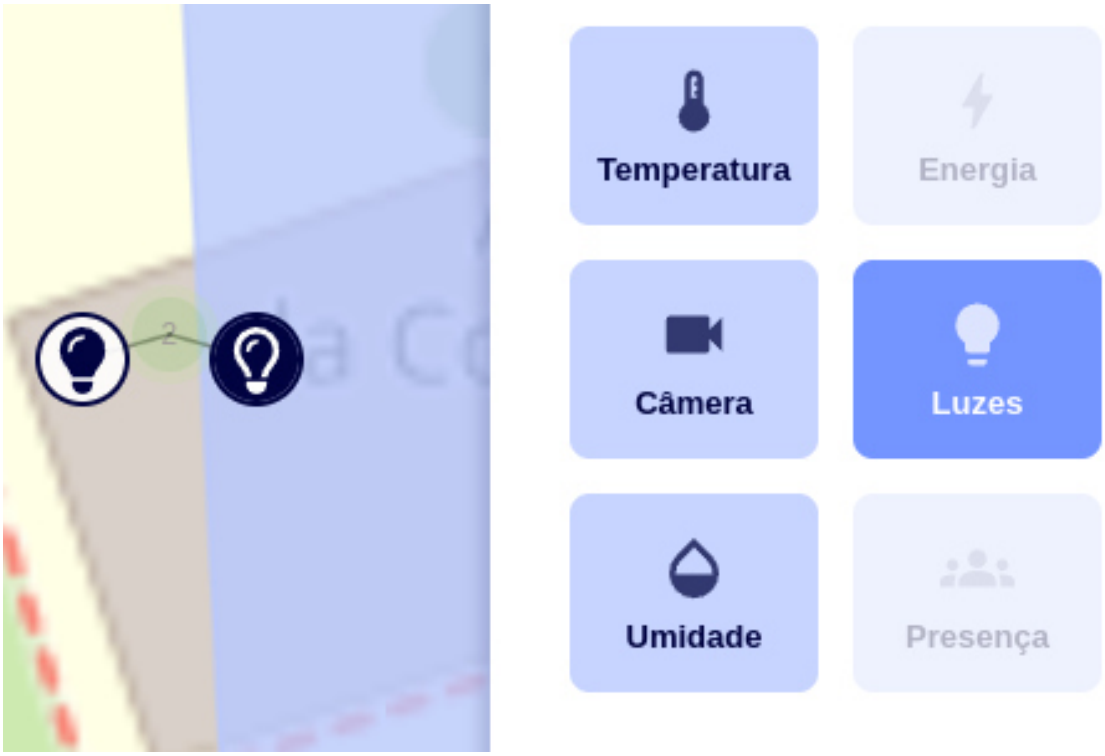
Figura 21 – Modal de confirmação da transição de estado de um atuador.



Fonte: Elaborada pelo autor

A implementação atual tem os dispositivos de Luzes como seu foco. Ao atualizar o estado do atuador, uma requisição é feita para transacionar o estado da luz, ligando-a ou desligando-a dependendo do seu estado atual, como ilustra a Figura 22.

Figura 22 – Exibição de dois atuadores em estados diferentes exibidos no mapa.

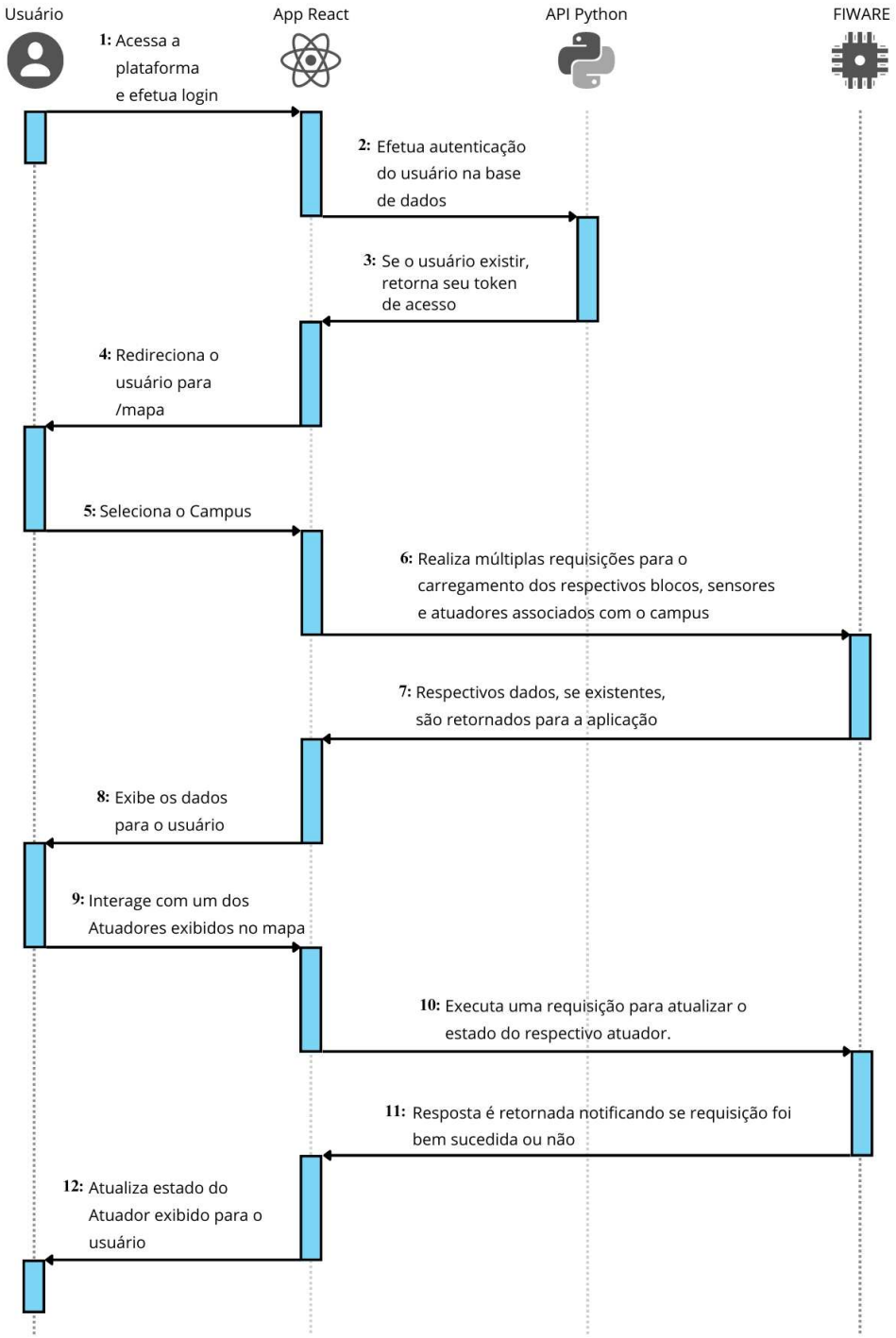


Fonte: Elaborada pelo autor

No entanto, a versão atual do projeto não verifica se houveram atualizações nos estados dos atuadores durante o uso da aplicação, seus respectivos estados são apenas atualizados no momento que o usuário recarrega a aplicação.

O diagrama da Figura 23 ilustra a sequência de requisições necessárias para a exibição e alteração do estado de um atuador:

Figura 23 – Diagrama de sequência das requisições de um atuador no mapa.



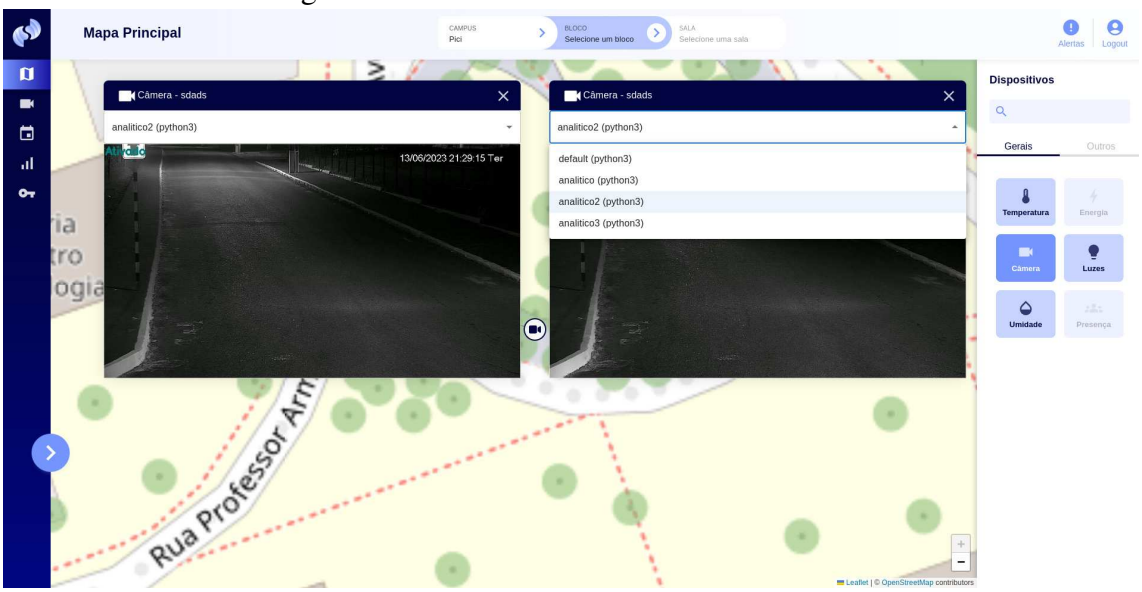
Fonte: Elaborada pelo autor

4.5.2.7 Componente CameraModel

Por fim, as câmeras possuem uma interação própria, já que seu comportamento é diferente dos observados nos sensores e nos atuadores. Ao clicar no ícone de uma câmera, é exibido o componente *CameraModal*, que mostra a transmissão de vídeo da respectiva câmera. O usuário ainda tem a possibilidade de alterar o modelo analítico de processamento de imagem aplicado à transmissão.

A Figura 24 ilustra o caso em que as duas instâncias do componente *CameraModal* exibem as imagens de suas respectivas câmeras, e mostra a funcionalidade de troca de modelo de processamento de imagem sendo aplicado à imagem da câmera:

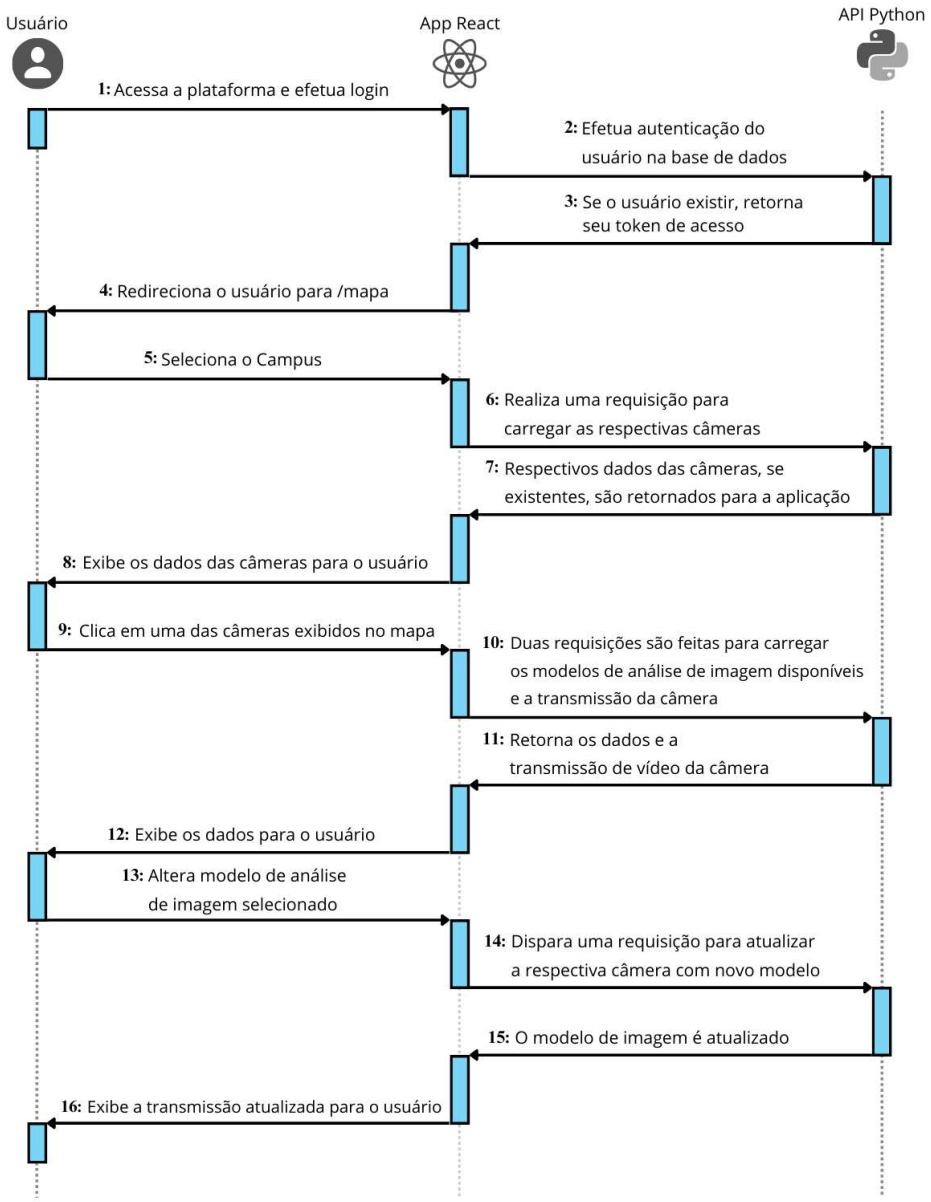
Figura 24 – Componente *CameraModal* com duas instâncias e troca de modelo de processamento de imagem



Fonte: Elaborada pelo autor

O diagrama da Figura 25 ilustra a sequência de requisições necessárias para a exibição e alteração dos dados das câmeras no componente:

Figura 25 – Diagrama de sequência das requisições do componente *CameraModal*



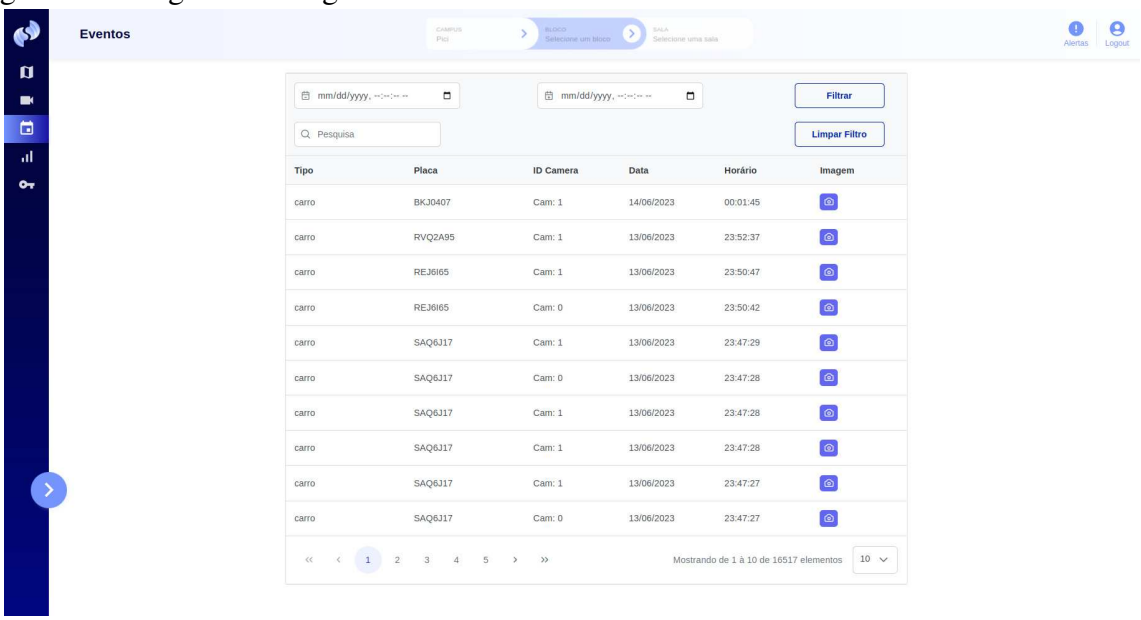
Fonte: Elaborada pelo autor

4.5.3 Tela de listagem de eventos

Na aplicação, os eventos são detecções observadas pelos modelos de processamento de imagem com base nas transmissões de vídeo das câmeras. Normalmente, um evento é a detecção de algum tipo de veículo com uma imagem do momento que o evento foi detectado.

Nesta página estão listados todos os eventos detectados até então, possibilitando ao usuário filtrá-los por meio de uma série de atributos, como mostra a Figura 26. Além disso, o usuário pode também visualizar dentro de um modal a imagem que está atrelada a cada evento específico.

Figura 26 – Página de listagem de eventos

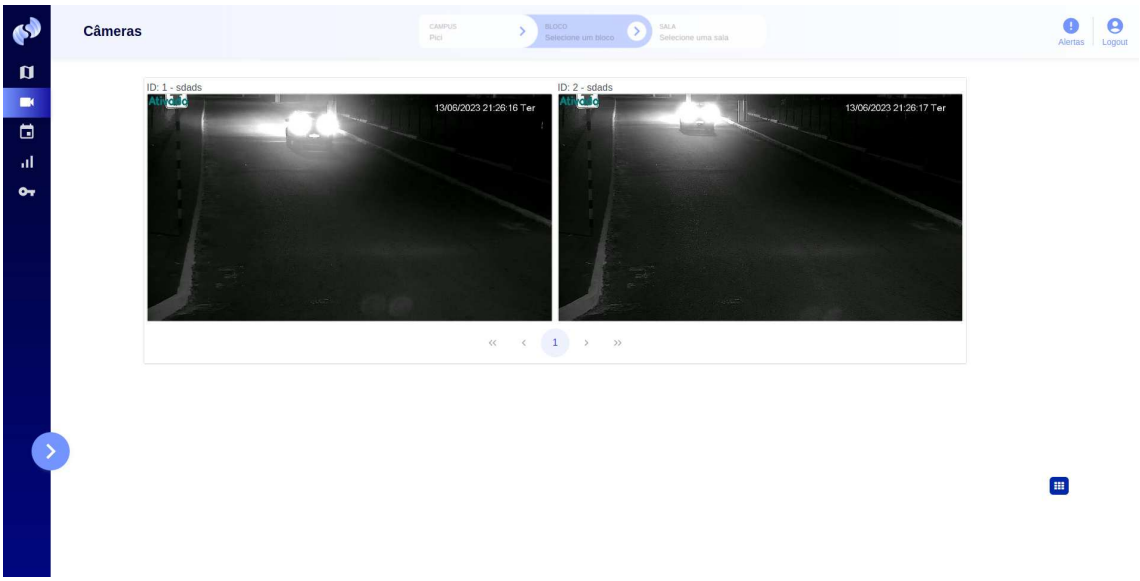


Fonte: Elaborada pelo autor

4.5.4 Tela de listagem de câmeras

Similar à tela de listagem de eventos, esta página exibe uma listagem com todas as câmeras e suas respectivas transmissões, permitindo ao usuário visualizar e interagir com todas elas simultaneamente, como mostra a Figura 27.

Figura 27 – Página de listagem de câmeras exibindo a transmissão de duas câmeras cadastradas no momento.



Fonte: Elaborada pelo autor

4.5.5 Acompanhamento da implementação

Ao longo da implementação, foi realizado um acompanhamento semanal com o gerente de projeto, a fim de monitorar e dar suporte durante todo o processo. Esse acompanhamento foi realizado através de chamadas online e por meio do compartilhamento de áudios e vídeos em uma plataforma de conversas.

Durante esse processo, foram feitas alterações pontuais na implementação de algumas funcionalidades. Isso ocorreu com a finalidade de facilitar o uso do usuário final ou se adequar a algumas limitações da arquitetura atual do *Smart Campus* deste trabalho.

4.6 Finalização da primeira versão do projeto

Nesta etapa, ocorre a finalização da primeira versão do projeto, pois já é possível constatar que todos os requisitos mínimos do projeto elencados durante o processo de planejamento do trabalho foram implementados, sendo necessário, no entanto, que estes ainda sejam testados e validados em uma próxima etapa de testes.

4.7 Realização de experimentos

Por fim, nesta etapa foram executados alguns experimentos a fim de medir a performance atual da aplicação, observando se o estado atual do projeto está de acordo com o esperado

para proporcionar uma experiência satisfatória aos usuários, além de identificar eventuais gargalos de performance a serem resolvidos em futuros desenvolvimentos do projeto. Para isso, cada tela da aplicação foi analisada com base em um conjunto de métricas utilizando a plataforma *Lighthouse*[10].

[10] https://developer.chrome.com/docs/lighthouse/overview/

5 EXPERIMENTOS REALIZADOS E DISCUSSÃO

Conforme mencionado até então, como o intuito do projeto é desenvolver uma interface web de fácil uso e que seja atrativa para um usuário utilizar, é importante que algumas métricas de performance sejam levadas em consideração durante o desenvolvimento.

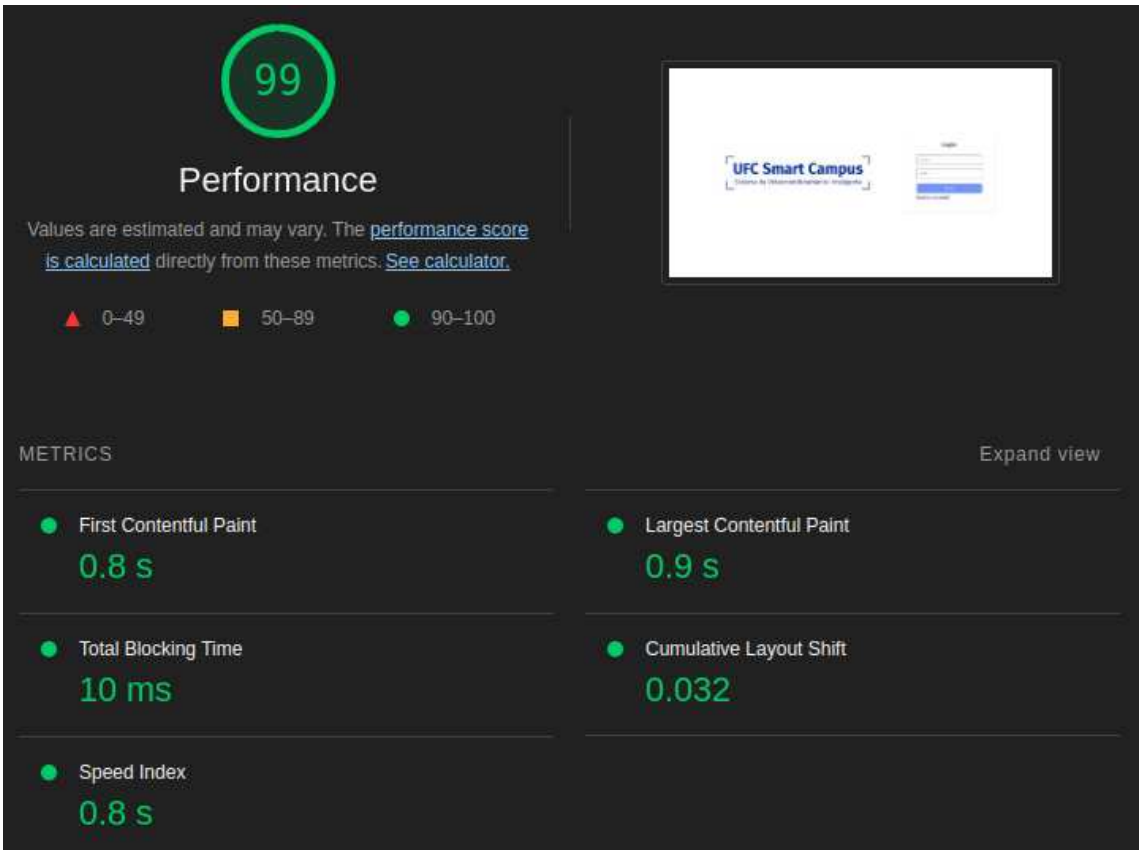
Neste capítulo, são apresentados os resultados experimentais da aplicação levando em consideração as seguintes métricas de performance na web:

- *First Contentful Paint - FCP* (Primeira renderização de conteúdo): é o tempo do começo do carregamento da página até a aplicação renderizar o primeiro conteúdo visível pelo usuário.
- *Largest Contentful Paint - LGC* (Maior renderização de conteúdo): é o tempo do começo do carregamento da página até a aplicação renderizar o maior conteúdo visível pelo usuário, sendo este o conteúdo que ocupa maior espaço na tela do usuário.
- *Total Blocking Time - TBT* (Tempo total de bloqueio): é o tempo entre o FCP e o momento em que a aplicação fica disponível para interação pelo usuário.
- *Cumulative Layout Shift - CLS* (Mudança cumulativa de layout): é uma métrica que lida com mudanças no layout exibido para o usuário após o conteúdo ter sido renderizado. Seu cálculo inicialmente levava em consideração todas as mudanças de layout depois que a aplicação terminou de carregar, no entanto, devido a mudanças recentes no desenvolvimento de aplicações web, agora o cálculo leva em consideração a maior soma de mudanças de layout em sequência dentro de um período de cinco segundos, em que o maior tempo entre mudanças é de um segundo.
- *Speed Index - SP* (Índice de velocidade): é uma métrica que lida com a velocidade com que o conteúdo é exibido na tela para o usuário.

As métricas acima foram capturadas utilizando a plataforma *Lighthouse* através da sua extensão para navegadores, com o projeto sendo executado localmente e em modo de produção. Assim, foi feita uma análise com base nos resultados experimentais obtidos para cada uma das principais telas da aplicação.

A Figura 28 apresenta os resultados para a tela de autenticação:

Figura 28 – Resultados do *Lighthouse* para a tela de autenticação com as métricas listadas anteriormente.



Fonte: Elaborada pelo autor

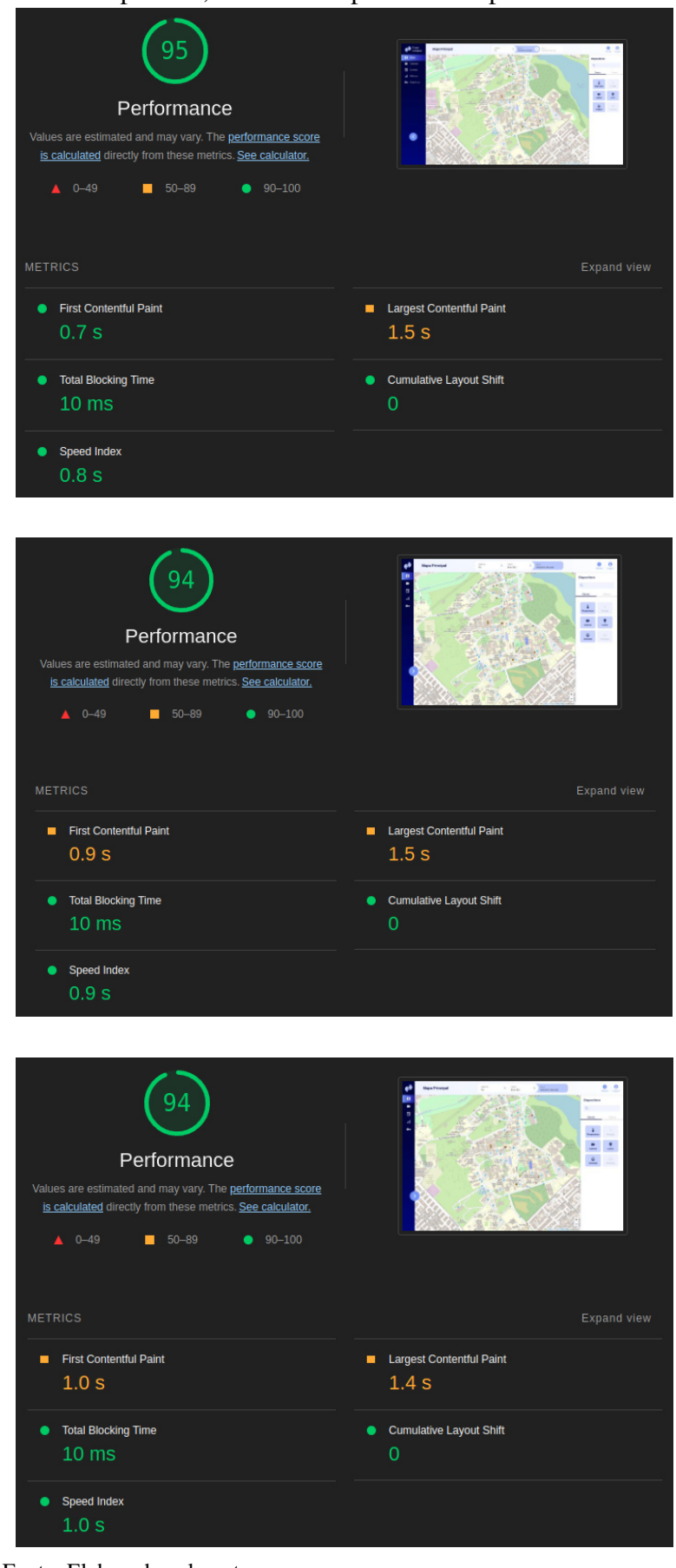
A tela de autenticação é bastante performática, até pelo fato de não possuir muito conteúdo a ser exibido. Tendo apenas um formulário com uma formatação bastante simples, o fato de possuir uma boa performance já é esperado.

Para a tela de mapeamento, foram analisados cenários com 16, 50 e 100 dispositivos, como mostra a Figura 29. Os dispositivos utilizados nos testes de 50 e 100 dispositivos foram virtuais, criados apenas com o intuito de testar a performance da aplicação.

Nesta tela é onde aparece a primeira métrica com nota mais baixa, o LCP. Isso se deve ao fato de todo o processamento e a exibição do mapa serem feitos pelo *Leaflet*, que carrega as imagens do mapa conforme o usuário interage com ele. Dessa forma, tanto é mais complexo de otimizar esse carregamento como acaba sendo um processo que se repete conforme o usuário navega pela aplicação.

Além disso, para os cenários com 50 e 100 dispositivos, pode-se observar um pequeno aumento nas métricas FCP e SI. Como nesses cenários há mais dados a serem carregados, a finalização da renderização dos primeiros elementos exibidos na tela do usuário é mais lenta, o que justifica a pequena piora nessas métricas.

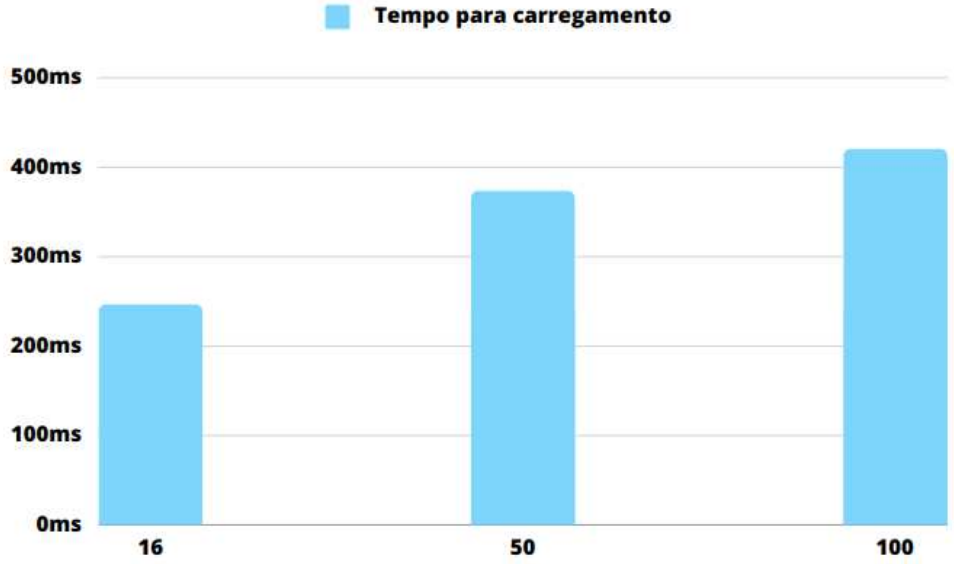
Figura 29 – Resultados do *Lighthouse* para a tela de mapeamento com as métricas listadas anteriormente para 16, 50 e 100 dispositivos respectivamente.



Fonte: Elaborada pelo autor

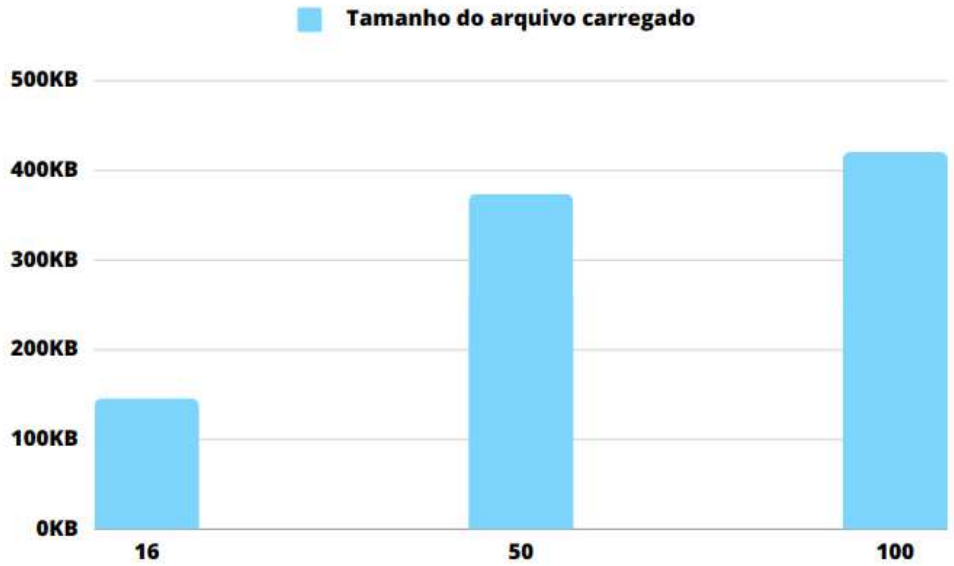
As Figuras 30 e 31 apresentam a evolução do tempo e do tamanho da requisição feita para carregamento de dispositivos na tela de mapeamento conforme a quantidade de dispositivos aumenta.

Figura 30 – Evolução do tempo de carregamento de dispositivos na tela de mapeamento.



Fonte: Elaborada pelo autor

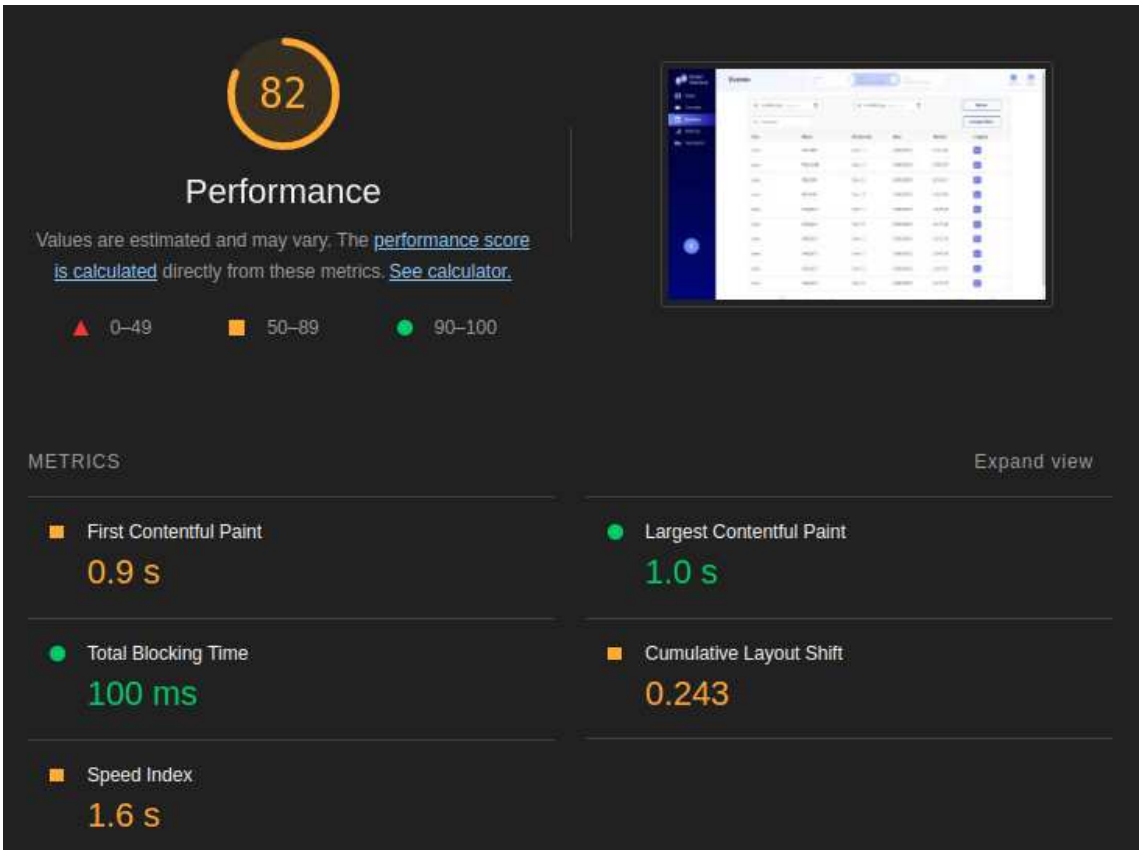
Figura 31 – Evolução do tamanho do arquivo de carregamento de dispositivos na tela de mapeamento.



Fonte: Elaborada pelo autor

A Figura 32 apresenta os resultados para a tela de listagem de eventos:

Figura 32 – Resultados do *Lighthouse* para a tela de listagem de eventos com as métricas listadas anteriormente.



Fonte: Elaborada pelo autor

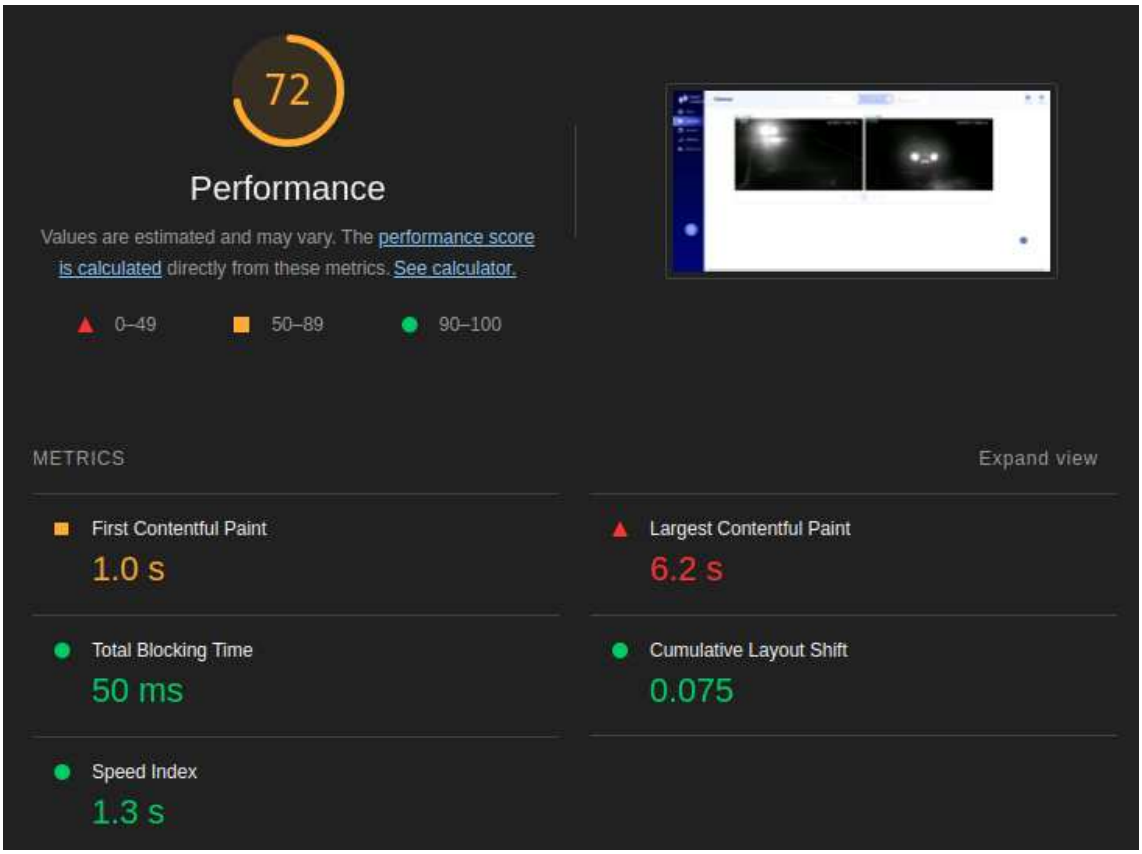
Na tela de eventos, apesar de também não haver muita complexidade em sua formatação, o fato dela depender dos dados carregados da API *Python* para exibir suas informações acarreta em uma performance mais baixa nas métricas FCP, CLS e SP.

Outro fator que impacta essa tela é que o carregamento da requisição de eventos, nos experimentos realizados, levou aproximadamente 5,94 segundos para completar. Isso se deve ao tamanho da requisição e da quantidade de eventos carregados de uma vez, 3.4MB e 19659, respectivamente, sendo valores excessivamente altos para serem realizados em apenas uma execução. Além disso, o design dessa tela não foi totalmente reformulado, o que leva a mais perdas de performance.

Um dos próximos passos deste trabalho são as correções desses problemas, sendo necessário atualizar o design da página para refletir o restante do projeto e, através da implementação de paginação ou cacheamento da requisição de eventos, reduzir a quantidade de dados carregados de uma só vez.

A Figura 33 apresenta os resultados para a tela de listagem de câmeras:

Figura 33 – Resultados do *Lighthouse* para a tela de listagem de câmeras com as métricas listadas anteriormente.



Fonte: Elaborada pelo autor

A tela de câmeras é bastante similar à de eventos, pois também necessita do carregamento de dados da API Python para poder exibir seus dados ao usuário. No entanto, como os dados exibidos ocupam um espaço considerável da tela do usuário, a métrica LCP é impactada. Além disso, seu design também não foi totalmente reformulado. Assim, durante os experimentos, a tela de listagem de câmeras levou 229ms para fazer o carregamento de 1,4kB de dados de duas câmeras.

Por fim, a Tabela 2 faz um compilado das informações sobre as diferentes requisições executadas nas principais telas da aplicação:

Tabela 2 – Resultados experimentais para as principais telas da aplicação.

Tipo de requisição	**Quantidade**	**Tempo de execução**	**Tamanho**
Dispositivos (sensores + atuadores)	16	247ms	14,6kB
Dispositivos (sensores + atuadores)	50	374ms	44,7kB
Dispositivos (sensores + atuadores)	100	421ms	88,4kB
Blocos acadêmicos	2	119ms	1,5kB
Salas de aula	2	123ms	1,3kB
Câmeras	2	229ms	1,4kB
Eventos	19659	5,94s	3,4MB
Imagem de um evento	-	471ms - 1,06s	18,3kB - 154kB

6 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Conforme discutido ao longo deste trabalho, a evolução e o desenvolvimento de cidades e campi inteligentes pode trazer contribuições massivas para a sociedade. No entanto, devido à sua complexidade e ao alto volume de dados envolvidos, seu uso pode ser comprometido pela dificuldade de visualização e acesso às informações coletadas. Assim, faz-se necessário o desenvolvimento de interfaces e visualizações cada vez mais otimizadas e com foco na experiência do usuário.

Nesse contexto, este trabalho teve como objetivo desenvolver uma aplicação web de uma interface para o Campus Inteligente da UFC, auxiliando no acesso e no uso das ferramentas e dados associados ao campus inteligente. Com isso, o trabalho buscou contribuir com discussões relevantes na área de *Smart City* e *Smart Campus*, além de contribuir também com potenciais melhorias no processo de tomada de decisões e de gestão dos campi e das cidades inteligentes.

A aplicação desenvolvida neste trabalho atende a todos os requisitos definidos na seção 3 do Capítulo 4 deste documento. Assim, após o fim do desenvolvimento das funcionalidades propostas, foram feitos alguns testes experimentais por meio da extensão para navegadores da plataforma Lighthouse, a fim de analisar a performance de cada uma das páginas da aplicação. Nos testes realizados, foram analisadas as métricas de FCP, LGC, TBT, CLS e SP.

De forma geral, após os testes com as métricas foi possível observar que os resultados obtidos foram satisfatórios. No entanto, algumas páginas da aplicação ainda necessitam de futuros ajustes para darem uma experiência ainda melhor para o usuário. Esses ajustes estão aqui listados como sugestão para desenvolvimento em trabalhos futuros:

- Ajustes e melhorias nas páginas de listagem de Eventos e Câmeras, para trazer a performance delas ao mesmo nível da aplicação como um todo. Além disso, deixa seu design mais próximo ao proposto pela designer.
- Revisão do código em geral da aplicação, que pode ser melhor estruturado para facilitar desenvolvimentos futuros, especialmente caso novos desenvolvedores entrem no projeto.
- Implementação e suporte a mais tipos de sensores e atuadores além dos descritos neste trabalho.
- Implementação de mapas para salas individuais, a fim de eventualmente dar suporte ao mapeamento de sensores e atuadores a nível de sala, além de apenas no mapa.

- Adição de um sistema de alertas para que o usuário seja avisado caso certas oscilações ocorram nas métricas observadas.
- Desenvolvimento de uma página de métricas para o usuário ter acesso a mais visualizações e dados históricos dos dispositivos que desejar.
- Integração com uma visualização do Grafana para o campus inteligente, caso o usuário queira um nível de visualização de dados mais granular do que a aplicação dispõe atualmente.

REFERÊNCIAS

ABUARQOUB, A.; ABUSAIMEH, H.; HAMMOUDEH, M.; ULIYAN, M.; ABU-HASHEM, M.; MURAD, S.; AL-JARRAH, M.; ALFAYEZ, F. A survey on internet of things enabled smart campus applications. In: . [S.l.: s.n.], 2017. p. 1–7.

AMURIM, A.; SILVA, J.; ORTIZ, M.; REGO, P.; SOUZA, J. Uma solução de iot baseada no fiware para gerenciamento de recursos energéticos e serviços acadêmicos em um campus universitário. In: **Anais do V Workshop de Computação Urbana**. Porto Alegre, RS, Brasil: SBC, 2021. p. 265–278. ISSN 2595-2706. Disponível em: <https://sol.sbc.org.br/index.php/courb/article/view/17119>.

BANDEIRA, L. K. R.; NETO, M. d. S. A. O que é um smart campus? **Perspectivas em Gestão & Conhecimento**, v. 12, n. 1, p. 175–188, abr. 2022. Disponível em: <https://periodicos.ufpb.br/ojs2/index.php/pgc/article/view/48610>.

BBCNEWS. **O dia em que a Terra vai atingir 8 bilhões de habitantes, segundo a ONU**. 2022. <https://www.bbc.com/portuguese/internacional-62067710>. Acesso em: 11 de Maio de 2023.

CHENG, B.; LONGO, S.; CIRILLO, F.; BAUER, M.; KOVACS, E. Building a big data platform for smart cities: Experience and lessons from santander (bigdata2015-5084). In: . [S.l.: s.n.], 2015.

DEEPAISARN, S.; YIWSIW, P.; TANTIWATTANAPAIBUL, C.; BUARUK, S.; SORNLERTLAMVANICH, V. Smart street light monitoring and visualization platform for campus management. In: **2022 17th International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP)**. [S.l.: s.n.], 2022. p. 1–5.

DONG, Z. Y.; ZHANG, Y.; YIP, C.; SWIFT, S.; BESWICK, K. Smart campus: definition, framework, technologies, and services. **IET Smart Cities**, v. 2, n. 1, p. 43–54, 2020. Disponível em: <https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-smc.2019.0072>.

FEITOSA, R. C. **Sistema de monitoramento para a área verde de um campus universitário inteligente**. Monografia (TCC) — Universidade Federal do Ceará, Campus do Pici, Fortaleza, 2022. Graduação em Engenharia da Computação.

FIWARE. **FIWARE Catalogue**. 2022. <https://www.fiware.org/catalogue/>. Acesso em: 12 de Junho de 2023.

GRAFANA. **Grafana**. 2019. <https://grafana.com/grafana/>. Acesso em: 15 de Junho de 2023.

IMD. **Smart City Index 2023**. 2023. <https://www.imd.org/smart-city-observatory/home/>. Acesso em: 3 de Junho de 2023.

INTELI. **Índice de Cidades Inteligentes - Portugal**. 2012. <https://docplayer.com.br/698315-Titulo-indice-de-cidades-inteligentes-portugal-organizacao-inteli-inteligencia-em-inovacao-centro-
html>. Acesso em: 11 de Maio de 2023.

JACOSKI, C. A.; HOFFMEISTER, L. M. Um modelo de campus inteligente para reorganização do ambiente universitário / a model of intelligent campus for reorganizing the university environment. **Brazilian Journal of Development**, v. 5, n. 2, p. 1373–1388, Jan. 2019. Disponível em: <https://ojs.brazilianjournals.com.br/ojs/index.php/BRJD/article/view/1111>.

LIN, Y.-B.; CHEN, L.-K.; SHIEH, M.-Z.; LIN, Y.-W.; YEN, T.-H. Campustalk: Iot devices and their interesting features on campus applications. **IEEE Access**, v. 6, p. 26036–26046, 2018.

MADAKAM, S.; RAMASWAMY, R.; TRIPATHI, S. Internet of things (iot): A literature review. **Journal of Computer and Communications**, v. 03, n. 05, p. 164–173, 2015.

MARTINI, P.; ROSAS, R. **Brasil tem 207,8 milhões de habitantes, mostra prévia do Censo 2022**. 2022. <https://valor.globo.com/brasil/noticia/2022/12/28/brasil-tem-2078-milhes-de-habitantes-mostra-prvia-do-censo-2022.ghtml>. Acesso em: 11 de Maio de 2023.

MOUHA, R. A. R. A. Internet of things (iot). **Journal of Data Analysis and Information Processing**, v. 09, n. 02, p. 77–101, 2021.

MUHAMAD, W.; KURNIAWAN, N. B.; SUHARDI; YAZID, S. Smart campus features, technologies, and applications: A systematic literature review. In: **2017 International Conference on Information Technology Systems and Innovation (ICITSI)**. [S.l.: s.n.], 2017. p. 384–391.

NAGOWAH, S. D.; STA, H. B.; GOBIN-RAHIMBUX, B. A. Towards achieving semantic interoperability in an iot-enabled smart campus. In: **2019 IEEE International Smart Cities Conference (ISC2)**. [S.l.: s.n.], 2019. p. 593–598.

NEGREIROS, I.; FRANCISCO, A. C. C.; FENGLER, F. H.; FARIA, G.; PINTO, L. G. P.; TOLOTTO, M.; ROGOSCHEWSKI, R. B.; ROMANO, R. R.; NETTO, R. S. Smart campus® as a living lab on sustainability indicators monitoring. In: **2020 IEEE International Smart Cities Conference (ISC2)**. [S.l.: s.n.], 2020. p. 1–5.

NEVES, A. R. de M.; SARMANHO, K. U.; JR., F. C. N.; MEIGUINS, B. S. Iniciativa smart campus: um estudo de caso em progresso na universidade federal do pará. In: **Anais do I Workshop de Computação Urbana**. Porto Alegre, RS, Brasil: SBC, 2017. ISSN 2595-2706. Disponível em: <https://sol.sbc.org.br/index.php/courb/article/view/2576>.

PAGLIARO, F.; MATTONI, B.; GUGLIERMENTI, F.; BISEGNA, F.; AZZARO, B.; TOMEI, F.; CATUCCI, S. A roadmap toward the development of sapienza smart campus. In: **2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC)**. [S.l.: s.n.], 2016. p. 1–6.

PALMA, D.; AGUDO, J.; SáNCHEZ, H.; MACíAS, M. An internet of things example: Classrooms access control over near field communication. **Sensors**, MDPI AG, v. 14, n. 4, p. 6998–7012, Apr 2014. ISSN 1424-8220. Disponível em: <http://dx.doi.org/10.3390/s140406998>.

PEREIRA, A. V. R. **Expandindo o Smartcampus: Um guia baseado em Storybook para desenvolvedores**. 2021. <http://dspace.sti.ufcg.edu.br:8080/xmlui/bitstream/handle/riufcg/19592/AMINTAS%20VICTOR%20RAMOS%20PEREIRA%20-%20TCC%20CI%c3%8aNCIA%20DA%20COMPUTA%c3%87%c3%83O%202021.pdf?sequence=1&isAllowed=y>. Acesso em: 09 de Junho de 2023.

PEREIRA, P. da S. **Desenvolvimento de um front-end web para uma ferramenta de experimentação em cenários de mobile cloud computing**. Monografia (TCC) — Universidade Federal do Ceará, Campus de Quixadá, Quixadá, 2022. Graduação em Sistemas de Informação.

PYPL. **PYPL PopularitY of Programming Language**. 2023. <https://pypl.github.io/PYPL.html>. Acesso em: 14 de Junho de 2023.

SASTRA, N. P.; WIHARTA, D. M. Environmental monitoring as an iot application in building smart campus of universitas udayana. In: **2016 International Conference on Smart Green Technology in Electrical and Information Systems (ICSGTEIS)**. [S.l.: s.n.], 2016. p. 85–88.

SINHA, S. **State of IoT 2023: Number of connected IoT devices growing 16% to 16.7 billion globally**. 2023. <https://iot-analytics.com/number-connected-iot-devices/>. Acesso em: 01 de Junho de 2023.

THINGSBOARD. **Thingsboard Dashboard**. 2023. <https://thingsboard.io/docs/user-guide/dashboards/>. Acesso em: 15 de Junho de 2023.

WIRECLOUD. **Wirecloud User Guide**. 2019. <https://wirecloud.readthedocs.io/en/stable/user_guide/>. Acesso em: 15 de Junho de 2023.

YANG, A.-M.; LI, S.-S.; REN, C.-H.; LIU, H.-X.; HAN, Y.; LIU, L. Situational awareness system in the smart campus. **IEEE Access**, v. 6, p. 63976–63986, 2018.

ZHAMANOV, A.; SAKHIYEVA, Z.; SULIYEV, R.; KALDYKULOVA, Z. Iot smart campus review and implementation of iot applications into education process of university. In: **2017 13th International Conference on Electronics, Computer and Computation (ICECCO)**. [S.l.: s.n.], 2017. p. 1–4.