

SmartStorage - Estoque inteligente para recebimento de encomendas

Carlos Eduardo dos Santos Junior, Dartagnan Scalon Machado, Lucas William Junges,
e Vinícius Henrique dos Santos

Abstract—SmartStorage é um sistema de gerenciamento de estoques inteligente para recebimento e armazenamento de encomendas pensado para contribuir na solução de problemas como o gerenciamento de encomendas entregues em condomínios residenciais, perda e furto de encomendas recebidas por terceiros, baixo detalhamento sobre a entrega e armazenamento incorreto das mesmas. Este artigo tem como objetivo descrever o desenvolvimento do sistema SmartStorage no seu âmbito de projeto-piloto, apresentando uma proposta para a implementação do software do sistema com foco no reconhecimento das etiquetas nas encomendas utilizando a biblioteca *Tesseract* de visão computacional e na comunicação entre o servidor, o banco de dados e o aplicativo. Os resultados obtidos foram a correta identificação e extração dos dados das etiquetas, o armazenamento dos dados pertinentes no banco de dados e a troca de informações entre o banco de dados e o aplicativo através do servidor. Após a implementação do aplicativo e do *back end* foi possível observar que o sistema se apresentou estável e funcional cumprindo com as propostas iniciais do projeto de forma satisfatória.

Palavras-chave: Estoque inteligente; Gerenciamento de encomendas; Visão computacional; *Tesseract*; *Back end*.

I. INTRODUÇÃO

O aumento crescente no *e-commerce* e *delivery* nos últimos anos vem exigindo cada vez mais um aprimoramento maior por parte dos condomínios quando se trata do gerenciamento de encomendas e correspondências. Diversas abordagens vem sendo utilizadas com o passar dos anos como, por exemplo, o uso de armários fechados para correspondência [5] porém, o grande desafio atualmente é encontrar uma solução acessível, prática e que possa acabar com a necessidade da mão de obra nas portarias dos condomínios, visto que as portarias digitais tem se tornado cada vez mais comuns dentre os condomínios residenciais do país [8] [9].

Segundo Andaros [3], com a pandemia da COVID-19, diversos síndicos afirmaram que o volume de movimentação nas portarias explodiu em apenas cinco meses, tendo em média um crescimento na entrega de encomendas e de *delivery* entre 40% e 70%. Sendo assim, a tendência é que o consumo de produtos através da internet se mantenha em alta, dada a adesão das pessoas por consumi-los desta forma e pelo fato de que ainda vive-se tempos de pandemia, onde existem apenas especulações de quando a situação se normalizará.

O sistema SmartStorage nasce como uma solução que busca oferecer um sistema composto por *hardware* e *software* que consiga atender a demanda dos condomínios tornando eficiente e simples o gerenciamento das encomendas que ali serão entregues facilitando a vida tanto dos moradores quanto dos

administradores, diminuindo a demanda por mão de obra, modernizando, conectando e promovendo a acessibilidade.

Assim, o projeto proposto versa sobre um sistema de armazenamento temporário de encomendas onde a encomenda à ser entregue é inserida em um compartimento instalado na entrada do condomínio. Dentro deste compartimento uma câmera é responsável por fotografar a etiqueta que contém as informações da encomenda e extrair da mesma as informações pertinentes, como nome do destinatário, apartamento, bloco, entre outras. Em seguida este módulo envia estas informações ao servidor, para que o mesmo consulte o banco de dados dos usuários do sistema e verifique à qual morador a encomenda em questão pertence. Após este processo, o morador identificado receberá um *e-mail* informando que sua encomenda foi entregue, e encontra-se armazenada na portaria do prédio.

De maneira a reforçar a relevância do tema abordado, estudos e projetos na área da otimização de estoques inteligentes têm sido amplamente realizados, como é o caso do projeto *An innovative automated storage and retrieval system for B2C e-commerce logistics* [6] onde foi desenvolvido um estudo de um novo tipo de estoque automatizado que utiliza um dispositivo de transporte multinível com acesso automático (IMCD) para organizar o estoque de acordo com a chegada das encomendas, baseando-se em um algoritmo *ant colony system* (ACS) para otimizar os caminhos que o dispositivo fará.

Outro estudo a ser mencionado é o *Design of Warehouse Control System For Automated Warehouse Environment* [11] onde o projeto aborda os sistemas já existentes de controle e gerenciamento de estoques apontando seus defeitos e sugerindo uma nova estrutura de controle. A estrutura de controle que foi proposta no artigo se chama *Smart ECS/WCS* onde os autores a descrevem como uma solucionadora dos pontos fracos presentes nos sistemas de controle de armazenamento já existentes. Apresentando uma dinâmica *PnP*, oferecendo uma variedade de funções e uma grande capacidade de armazenamento e coleta de dados, a *Smart ECS/WCS* mostrou ser uma poderosa ferramenta para aprimorar a eficiência da operação de estoques.

O Grande diferencial do sistema SmartStorage é que ele é uma solução completa, incluindo o *hardware* e o *software*, podendo assim solucionar tanto o problema de gerenciamento de estoque quanto o de automatização do recebimento das encomendas. É uma maneira inovadora de automatizar a entrega de encomendas nos condomínios residenciais dispensando a mão de obra humana no momento do recebimento e armazenamento e podendo assim, oferecer mais praticidade e segurança aos moradores.

O trabalho aqui apresentado teve como objetivo o desenvolvimento e implementação do aplicativo e dos softwares que compõem o sistema SmartStorage, com o foco na conectividade e troca de informações entre eles. Desta maneira, o projeto descrito neste artigo foi considerado em um contexto simplificado de modo que o desenvolvimento e implementação do hardware, devido ao atual panorama mundial, não fazem parte deste projeto-piloto.

Sendo assim, na seção II do artigo são apresentadas informações detalhadas sobre o sistema SmartStorage e como o mesmo é composto. Na seção III é delimitado o contexto do projeto-piloto e o cenário de simulação utilizado. Na seção IV, são apresentadas as partes que compõem o projeto-piloto. Na seção V os resultados obtidos são apresentados e por fim, na seção VI é apresentada a conclusão do trabalho e os comentários sobre quais são os próximos passos necessários para dar continuidade no projeto.

II. SMARTSTORAGE

SmartStorage é um sistema completo de recebimento e armazenamento de encomendas pensado especialmente para atender as necessidades de condomínios residenciais. Por se tratar da idealização de uma solução completa o sistema é composto por hardware e software e seus principais componentes são: banco de dados na nuvem para os usuários, banco de dados local para as encomendas, servidor, aplicativo e o módulo de recebimento de encomendas.

Todo o desenvolvimento do sistema foi pensado para oferecer uma maior comodidade ao usuário, com a maior qualidade possível de serviço, garantindo o funcionamento pleno e correto do sistema através da utilização de tecnologias atuais como o armazenamento de dados na nuvem e hospedagem do algoritmo de reconhecimento de etiquetas no servidor, tornando o aplicativo desenvolvido mais leve e mais otimizado para o uso.

Um dos objetivos do sistema é que o mesmo seja de fácil utilização, sendo assim, o aplicativo desenvolvido permite que o usuário crie um cadastro próprio e tenha acesso às suas informações cadastradas. É no aplicativo também que o usuário poderá visualizar as suas encomendas entregues e onde as mesmas estão armazenadas, para que assim possa fazer a retirada.

Para garantir que o aplicativo esteja disponível para um maior número de usuários, independente do seu dispositivo Android, e para uma maior comodidade dos mesmos a maior parte do processamento é realizado no servidor. Desta forma o aplicativo se torna apenas um canal pelo qual o usuário visualiza as informações sobre as suas encomendas. Para isto foi implementado um *Back end* na linguagem *Python* através do uso do *framework Flask* responsável por implementar as rotas através das quais as informações necessárias serão trocadas pelos módulos do sistema. O servidor atuará como um integrador do projeto, sendo responsável por conectar o banco de dados na nuvem do aplicativo ao banco de dados local das encomendas, permitindo a consulta dos dados lá armazenados no momento do reconhecimento das etiquetas pelo algoritmo de visão e posteriormente os enviando para a interface homem-máquina (IHM).

Para verificar à quem pertence a encomenda que está sendo recebida pelo sistema, a câmera instalada no compartimento de recebimento terá a função de registrar uma fotografia da etiqueta contendo as informações referentes ao destinatário daquela encomenda. Esta fotografia, será então enviada ao servidor onde se encontra o algoritmo de visão computacional. O algoritmo de visão computacional foi implementado utilizando o *software Tesseract*. Este algoritmo será responsável por extrair as informações da etiqueta e disponibilizá-las de volta ao servidor para que seja feita a consulta no banco de dados local e verifique-se a qual usuário pertence a encomenda entregue. Para uma maior segurança do sistema e para que ocorra a correta notificação do usuário, ao serem identificadas as informações sobre o destinatário da encomenda as mesmas serão mostradas na IHM em um monitor ao lado do compartimento de recebimento. Desta forma o entregador será responsável pela verificação das informações da entrega, tendo a possibilidade de alterar algum dado caso o sistema não o tenha reconhecido corretamente, e de confirmar então a entrega. Ao ser confirmada a entrega o servidor se encarrega de notificar o destinatário via *e-mail* sobre o recebimento.

O projeto que foi desenvolvido se encaixa na fase de projeto-piloto, onde todas as partes que compõem o software do sistema foram definidas conceitualmente e foi realizada uma implementação base das principais funcionalidades. Devido ao escopo deste trabalho, neste momento a definição do *hardware* e *software* que compõe o módulo de recebimento das encomendas e do módulo de estoque automatizado serão realizados em projetos futuros. A Figura 1 ilustra a logomarca criada para o sistema em desenvolvimento, que busca gerar identificação com os serviços atuais de entrega de forma a aumentar a aceitação e facilitar a adaptação ao uso do sistema.



Fig. 1: Logomarca SmartStorage

III. CENÁRIO DE SIMULAÇÃO E PROJETO-PILOTO

A simulação do sistema delimitado no projeto-piloto tem como foco o desenvolvimento e a integração das partes que compõem o software do sistema, portanto foram implementadas a identificação ótica de uma encomenda e a comunicação entre o aplicativo, bancos de dados e o servidor.

O principal objetivo da simulação consiste em fazer o correto reconhecimento de uma encomenda e a notificação ao usuário sem necessitar do sistema físico. Sendo assim, as etiquetas que o algoritmo de visão computacional reconhecerá não passarão pelo procedimento de serem fotografadas pela câmera e já estarão previamente disponíveis no computador

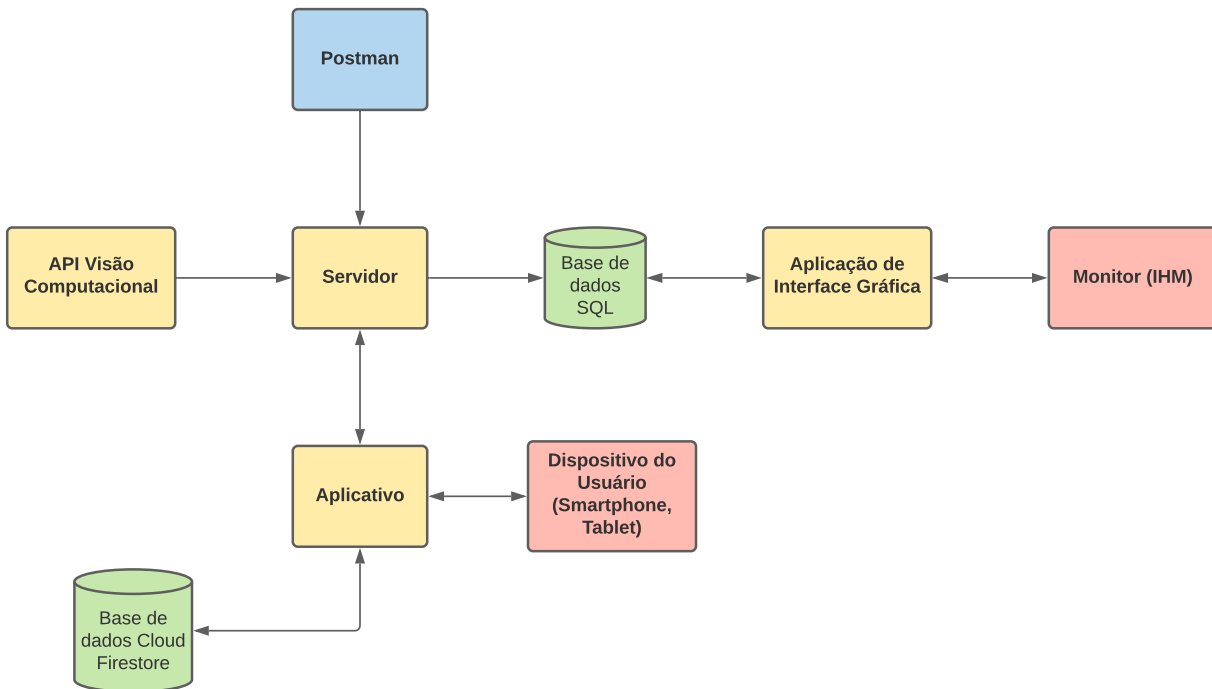


Fig. 2: Fluxograma de dados

do servidor e serão processadas e reconhecidas conforme requisição enviada para o mesmo em formato JSON. Ao receber essa informação, o servidor se encarregará de executar o algoritmo de visão computacional que prosseguirá para a extração das informações da imagem. Uma vez que as informações forem extraídas, o algoritmo as envia ao servidor para que o mesmo armazene-as no banco de dados fazendo a relação com o destinatário da imagem. O fluxo dos dados pode ser visto na Figura 2 acima.

As etiquetas utilizadas para a simulação foram geradas através do gerador padrão de etiquetas dos Correios (disponível no site dos Correios), de forma a garantir que o padrão utilizado para o projeto se assemelhe o máximo possível a uma situação real. A Figura 3 ilustra uma das etiquetas geradas que foi utilizada nas simulações.



Fig. 3: Exemplo de etiqueta utilizada nas simulações

IV. SOFTWARE DO SISTEMA

A. Aplicativo SmartStorage

Para o desenvolvimento do aplicativo optou-se pelo ambiente de desenvolvimento (*IDE*) *Android Studio* que utiliza a linguagem Kotlin e Java para a programação. Apesar de permitir a criação e execução do aplicativo apenas para sistemas operacionais Android, a escolha desta ferramenta para o desenvolvimento é justificada e foi motivada pelo fato de que este software oferece diversas Interfaces de Programação de Aplicação (*API's*) já integradas em sua plataforma, além de outras ferramentas como o banco de dados Firebase e um simulador de dispositivos Android, tornando mais simples a integração e utilização destas ferramentas com o aplicativo e facilitando a implementação e depuração do código.

O objetivo da implementação do aplicativo é conectar o usuário aos bancos de dados e ao servidor, permitindo assim que este receba uma notificação em seu celular no momento da entrega de sua encomenda. Sendo assim, o aplicativo deve ser capaz de enviar os dados cadastrais referentes às informações pessoais do usuário ao servidor para que este as armazene em seu banco de dados local. No momento da entrega e reconhecimento de uma encomenda, o aplicativo deve receber as informações do servidor de que a encomenda referente a um determinado usuário foi entregue e assim, mostrar a este usuário no aplicativo que existe uma ou mais encomendas disponíveis para retirada.

B. Banco de Dados

Dois bancos de dados foram utilizados para o desenvolvimento do projeto. O *Cloud Firestore* é um banco de dados

relacional SQL.

1) *Cloud Firestore*: O *Cloud Firestore* é um banco de dados na nuvem que foi utilizado para guardar as informações referentes ao login e o cadastro dos usuários. A utilização deste banco de dados para estas funções se deu pelo fato de que o *Android Studio* dentro de sua plataforma consegue integrar o banco de dados do *Cloud Firestore* de maneira muito simples e possui diversas funções já prontas facilitando a implementação e desenvolvimento do aplicativo.

2) *Banco de Dados Relacional*: O banco de dados relacional com tecnologia SQL foi criado para o servidor a fim de armazenar e ser utilizado para gerir as informações trafegadas pelo mesmo. Para isto foi utilizada a biblioteca *Peewee* do *Python*, uma biblioteca de mapeamento objeto-relacional (ORM) prática e funcional para gerenciar banco de dados.

C. Servidor

Para o servidor será necessária a implementação de um *Backend*. Utilizando a linguagem *Python* e através do *framework Flask*, o mesmo será responsável por criar as rotas através das quais as informações necessárias serão trocadas pelos módulos do sistema. O servidor atuará como um integrador do projeto, sendo responsável por conectar o banco de dados na nuvem do aplicativo ao banco de dados local das encomendas, permitindo a consulta dos dados lá armazenados no momento do reconhecimento das etiquetas pelo algoritmo de visão e posteriormente os enviando para a IHM.

D. Reconhecimento de Etiquetas

Para o reconhecimento de etiquetas, optou-se por utilizar o algoritmo de visão computacional chamado *Tesseract*, uma biblioteca muito robusta baseada em redes neurais. Para que o sistema de leitura funcione corretamente, primeiramente efetuaram-se testes com as etiquetas para a parametrização da biblioteca de forma que ela consiga ler com boa confiabilidade etiquetas de aproximadamente 400 x 300 pixels de tamanho. Em adição a isto também é realizada a mineiração das linhas de texto, retirando caracteres especiais e quebras de linha indesejados. Desta forma é possível obter um resultado preciso para a leitura, aproveitando-se da padronização existente entre as etiquetas

E. IHM

Para este projeto, foi criado também um *software* para interagir com o entregador da encomenda conforme ilustrado na figura 4 abaixo.

A interface de usuário foi criada de maneira a ter um aspecto visualmente agradável utilizando-se a tecnologia *TKinter* do *Python*. Abaixo são descritas funcionalidades da interface gráfica:

- Botão Atualizar - Quando o entregador acionar este comando, o sistema irá atualizar a tela com os dados da última compra lida pelo módulo de recebimento de encomendas;

- Botão Alterar - Quando pressionado irá atualizar os dados no banco de dados do servidor, garantindo o correto reconhecimento das informações, atualizando estas para o futuro;
- Botão Confirmar dados - Este botão dispara uma série de eventos internos no servidor ao ser pressionado. Estes eventos são responsáveis desde enviar um email ao destinatário da encomenda até preparar o sistema para o recebimento da próxima encomenda.

A biblioteca *SMTPLib* foi utilizada para a configuração e envio dos *e-mails* responsáveis pelas notificações. Esta interface pode ser executada em qualquer dispositivo que possua uma arquitetura x86 ou ARM e que seja capaz de se comunicar com o banco de dados do servidor.

V. RESULTADOS

Nesta seção serão apresentados os resultados que foram obtidos para cada parte que compõe o conjunto referente ao software do sistema *SmartStorage*.

A. Aplicativo *SmartStorage*

O aplicativo [4] que foi desenvolvido é composto por 5 telas sendo estas: *login*, cadastro, menu, minhas encomendas e minhas informações. O mesmo permite que o usuário navegue entre estas telas tendo acesso as encomendas que foram entregues e as suas informações cadastradas.

Além disso, o desenvolvimento do aplicativo foi pensado para garantir a segurança e exclusividade do uso por seus usuários, onde foi implementado um cadastro personalizado para que novos usuários criem sua conta e tenham um login próprio com e-mail e senha de acesso ao aplicativo, podendo assim garantir que terceiros não tenham acesso às suas informações pessoais e suas encomendas. Caso algum usuário esqueça sua senha cadastrada, o aplicativo oferece a opção "esqueci minha senha", onde o usuário informa seu e-mail cadastrado e caso o mesmo esteja presente no banco de dados, será enviada uma mensagem para este e-mail com um link para a criação de uma nova senha.

Ao efetuar o login, o usuário é redirecionado ao menu principal onde ali poderá navegar entre as funções disponíveis no aplicativo ou efetuar o logout pressionando o botão "sair".

Na tela "Minhas Encomendas" irão aparecer as encomendas disponíveis para serem retiradas e o número do *box* em que se encontram. O aplicativo receberá do servidor estas informações e as exibirá para o usuário. O usuário poderá então retirar o seu pacote ao pressionar o botão "Retirar Encomenda" e o aplicativo se encarregará de enviar para o servidor que a encomenda está pronta para ser retirada e o mesmo fará o processamento deste pedido marcando-a como "Retirada" dentro do banco de dados.

Em sua versão atual, o aplicativo possui capacidade de mostrar 5 encomendas ao mesmo tempo, sendo que o botão "Retirar Encomenda" será visível somente se alguma encomenda estiver pronta para ser retirada. Caso contrário, este botão permanece invisível. Caso haja mais do que 5 encomendas, ao retirar-se uma encomenda a tela é atualizada mostrando o próximo pacote da lista.



Smart Storage - Receptor de Encomendas

Os dados do favorecido serão exibidos nos campos abaixo!

Nome

Endereço

Bloco AP

A última encomenda lida será carregada, clique em Atualizar!

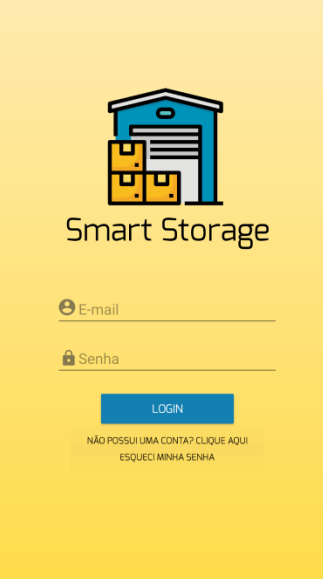
Atualizar **Alterar** **Confirmar Dados**

Smart Storage

Produto sem foto

Fig. 4: Interface de usuário criada para o projeto

O design do aplicativo e as telas que compõem o mesmo podem ser visualizadas nas Figuras 5,6,7,8 e 9 que correspondem à tela de login, cadastro, menu, minhas encomendas e minhas informações respectivamente.



Smart Storage

E-mail

Senha

LOGIN

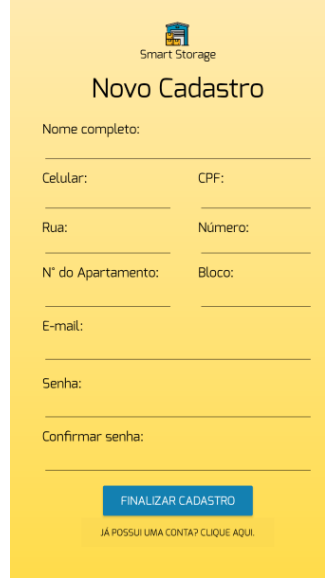
NÃO POSSUI UMA CONTA? CLIQUE AQUI

ESQUECI MINHA SENHA

Fig. 5: Login

B. Banco de Dados

1) *Cloud Firestore*: O armazenamento das informações de login foi feito utilizando um serviço específico de autenticação em uma sessão chamada *Authentication* do *Cloud Firestore* onde é possível verificar diversas informações com relação ao número de novos cadastros, número de *logins* e *logouts* assim como os *e-mails* cadastrados no aplicativo como mostra a Figura 10. Dentro da sessão *Authentication* também que foi



Novo Cadastro

Nome completo:

Celular: CPF:

Rua: Número:

Nº do Apartamento: Bloco:

E-mail:

Senha:

Confirmar senha:

FINALIZAR CADASTRO

JÁ POSSUI UMA CONTA? CLIQUE AQUI

Fig. 6: Cadastro

definido o *template* do *e-mail* que é enviado ao usuário no caso de recuperação de senha como ilustra a Figura 11.

O armazenamento dos demais dados cadastrais dos usuários fica contido na sessão *Firestore*, que nada mais é que um serviço de armazenamento na nuvem. Ali, foi criado um documento chamado *usuarios* que armazena todas as informações pessoais, as organizando de acordo com o *User ID* de cada usuário, gerada automaticamente como ilustra a Figura 12.

2) *Banco de Dados Relacional*: O banco de dados relacional foi criado utilizando a biblioteca *Peewee* conforme dito anteriormente. A Figura 13 ilustra as classes instanciadas no banco de dados do servidor.



Fig. 7: Menu

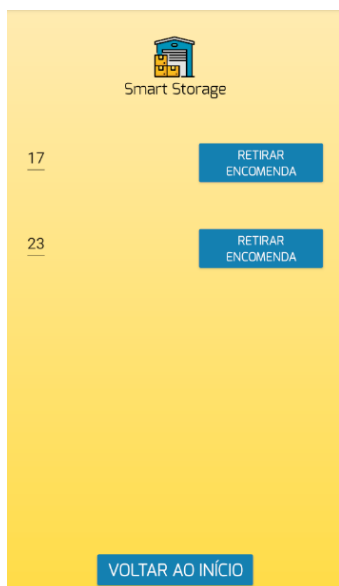


Fig. 8: Minhas Encomendas

C. Servidor

Para realizar a comunicação entre todas as aplicações utilizou-se o protocolo *http* onde o servidor foi implementado através do *framework Flask* em linguagem de programação *Python*. Para realizar as comunicações, as rotas que realizam as requisições do tipo *GET* e *POST* foram implementadas com uma *API Restfull*. A lógica do fluxo de dados pode ser vista na Figura 2.

Para validação das rotas e teste do *Back end* foi utilizado o *software Postman* que é uma API destinada à realizar requisições *web* personalizadas. Assim, para a simulação da comunicação com o servidor o *software* foi conectado à rede local na porta utilizada pelo *Flask*. Como a criação do módulo de recebimento não estava no escopo deste projeto-piloto, as



Fig. 9: Minhas informações

etiquetas utilizadas foram armazenadas em um diretório da máquina do servidor. Posteriormente, quando o módulo de recebimento for implementado a única diferença será a origem da imagem, que será captada pela câmera e enviada pela rede até o servidor.

A comunicação do servidor com a aplicação gráfica foi realizada através do banco de dados relacional, ou seja, após o servidor armazenar as informações no banco de dados relacional a interface visual consulta o mesmo exibindo ao entregador as informações referentes ao destinatário.

Foram criadas as seguintes rotas de conexão do servidor com o aplicativo:

- Rota de cadastro - O servidor recebe os dados cadastrais do aplicativo e automaticamente irá salvá-las para o posterior uso pela interface gráfica (IHM);
- Rota de exclusão - Quando o usuário selecionar no aplicativo a opção "Retirar encomenda", o aplicativo manda este aviso ao servidor que faz a exclusão da encomenda em questão do banco de dados;
- Rota de encomendas - O aplicativo solicita ao servidor para que o mesmo envie à ele as informações de encomendas de um dado usuário. Caso haja alguma encomenda para este usuário aguardando retirada a mesma será mostrada no aplicativo.

A figura 14, na página 8, contém a ilustração do servidor em funcionamento na porta 5000 da rede local.

D. Reconhecimento de Etiquetas

Para reconhecimento das etiquetas primeiramente foi definido um diretório padrão o armazenamento das figuras. O software de reconhecimento de caracteres sempre irá buscá-las neste local e quando a captura de imagem for realizada esta imagem deverá ser um arquivo com a extensão *.png* pois este é o tipo de imagem que melhor se adaptou a tecnologia Tkinter durante os testes do aplicativo. Ao receber o nome da etiqueta, o software irá buscá-la no diretório padrão, executará

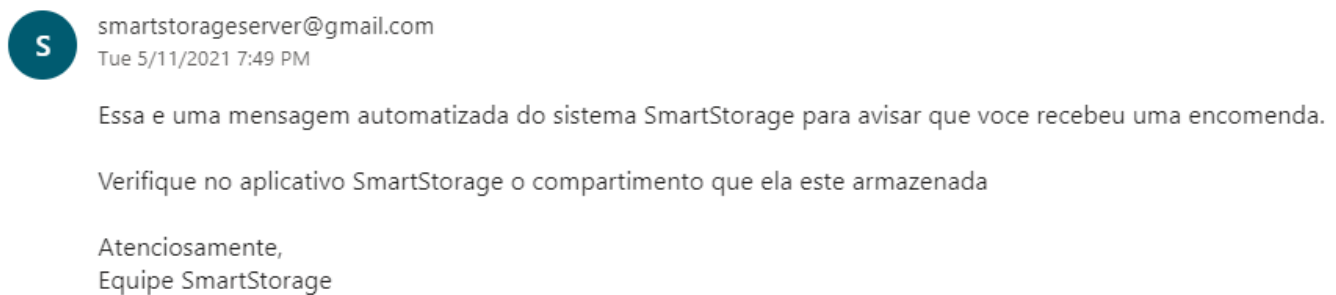
SmartStorageApp ▾

Authentication

Users Sign-in method Templates Usage

✦ Prototype and test end-to-end with the Local Emulator Suite, now with Firebase Authentication [Get started](#)

Identifier	Providers	Created	Signed In	User UID ↑
dscalon@live.com	✉	May 4, 2021	May 9, 2021	218jru0MxkXVAj3J378A4wi779S2
asdc@gmail.com	✉	May 8, 2021	May 8, 2021	3kA2o0BoNZVa6Zbe8VaCBxdwLj...
vinicius.h.santos31@gmail...	✉	Apr 27, 2021	May 8, 2021	5q5JBar2S1VQ5eECeFLWBUoG6I2
vinicius@gmail.com	✉	Mar 30, 2021	Apr 27, 2021	9fYVdmI8JIN51Rlnnn0sbwfskpF2
jorge.da.silva@gmail.com	✉	May 6, 2021	May 6, 2021	H4cP043UR8cbeQC7T2a3p9i3Bft2

Fig. 10: Armazenamento dos *logins*Fig. 11: *Template* do e-mail de recuperação de senha

🏠 > usuarios > zG0uVSM3pPVL...

smartstorageapp-26984	usuarios	zG0uVSM3pPVLXkdQlprjsGXJV12
+ Start collection	+ Add document	+ Start collection
usuarios >	eK0NuvPLRfcqH6Iwbq8Gp0NqFIz2 fhdy9N3PveaJEEqbN13PNyNUzJ02 fqb7p8D07VfHr7yIWw9ndW8RSD52 gZIf84PZUebVRKko0ygg4qX6mCo2 hF08fYvNXTOV5kvMKNUuFX2uS0g1 hu1xVKGfCfLXc1QKibK6h0ZKumo22 ibQ29hPmpFV20cBs5E95Aj3pQN83 kfz0vCFzDgVdKasPmnWTOhQi1Mv1 1sj3cNuB1HUJuTpZgrnrsiXqaF63	+ Add field apto: "101" bloco: "A" celular: "4798765432" cpf: "98765432101" email: "projetoespecializadofsc@gmail.com" nome: "Projeto Especializado" numero: "123" rua: "João Pessoa"

Fig. 12: Armazenamento dos cadastros

o algoritmo de visão e retornará o usuário lido, conforme ilustrado na figura 15 apresentada na página seguinte

O status 200, apresentado na figura 15 indica que o processamento e envio das informações para o servidor foi realizado com sucesso.

E. IHM

Após o processo de cadastramento, e de leitura de etiqueta, ao atualizar os dados do usuário da IHM terá um resultado semelhante ao da figura 16 abaixo.

Conforme mostrado na sub-seção anterior após o reconhecimento do nome do receptor da encomenda, o software de visão computacional transmitiu o nome do destinatário

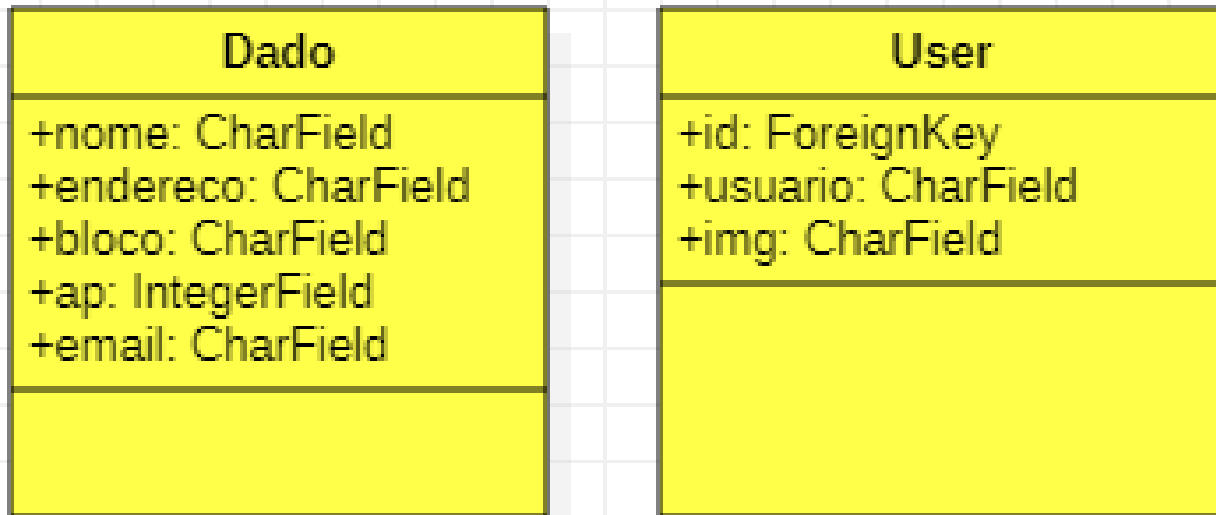


Fig. 13: Classes instanciadas para o Banco de dados relacional

```

ca. Prompt de Comando - python Servidor.py
* Serving Flask app "Server Smart Storage" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
  
```

Fig. 14: Servidor criado para o projeto

ao servidor. Assim, ao ser clicado no botão "Atualizar" da interface gráfica, a mesma busca no banco de dados do servidor as informações referentes àquele usuário. Caso o usuário não esteja cadastrado a interface exibirá os campos em branco, mostrando que o usuário ainda não possui cadastro no sistema. Ainda assim, será possível que o entregador ou porteiro do condomínio realize um pré-cadastro de usuário no sistema, através do botão alterar.

Ao confirmar os dados, o sistema irá alocar um *box* para armazenar a encomenda e irá enviar um email padronizado, como demonstrado na figura 17 abaixo.

VI. CONCLUSÃO

SmartStorage é um sistema de gerenciamento de estoques inteligente para recebimento e armazenamento de encomendas

pensado para contribuir na solução de problemas relacionados a entrega de encomendas em condomínios residenciais, perda e furto de encomendas recebidas por terceiros, baixo detalhamento sobre a entrega e manuseio e armazenamento incorreto das mesmas.

O sistema SmartStorage, quando concluído, tratará de uma solução completa, incluindo o *hardware* e o *software*, podendo assim solucionar tanto o problema de gerenciamento de estoque quanto o de automatização do recebimento das encomendas. É uma maneira inovadora de automatizar a entrega de encomendas nos condomínios residenciais dispensando a mão de obra humana no momento do recebimento e armazenamento e podendo assim, oferecer mais praticidade e segurança aos moradores.

Os principais resultados e contribuições obtidas com a execução do projeto foram o desenvolvimento e

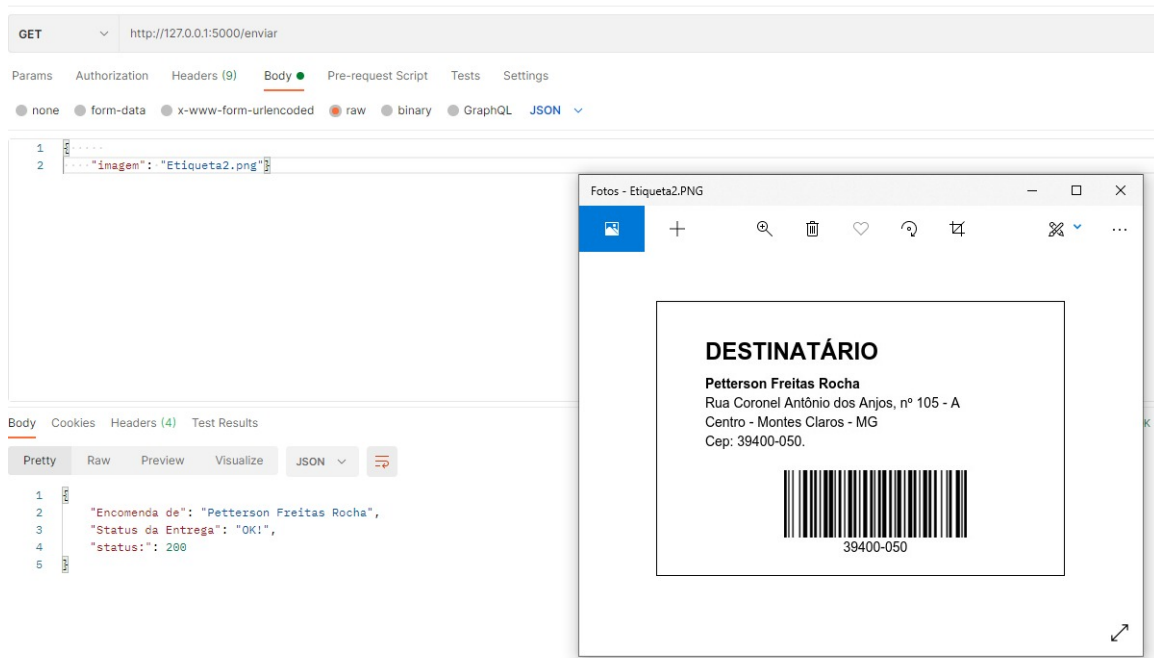


Fig. 15: Leitura e extração dos dados da etiqueta



Fig. 16: Interface de usuário

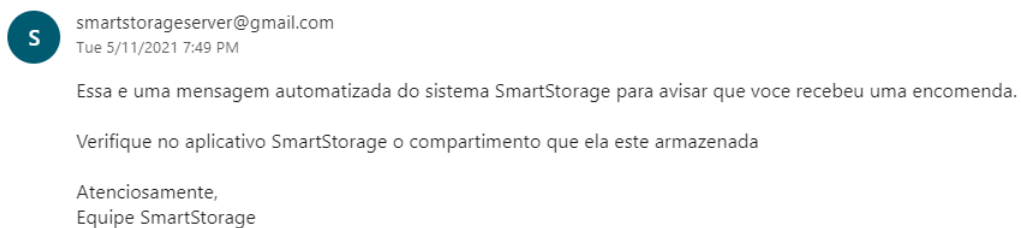


Fig. 17: Email enviado ao usuário

implementação do servidor e da IHM, a utilização do algoritmo de visão computacional para extrair os dados da etiqueta, o uso do servidor para consultar os bancos de dados a fim de verificar à quem pertence a encomenda, a criação de uma IHM, do aplicativo destinado ao usuário e por fim, a comunicação entre os softwares feita pelo servidor. O

grupo acredita ter dado um grande passo em direção a um futuro mais inclusivo e seguro, levando praticidade as pessoas através do uso da engenharia.

Considerando que a execução do projeto se deu na forma de um "projeto-piloto", dado o fato da limitação de tempo e da adaptação aos recursos disponíveis, acredita-se que conseguiu-

se atingir o principal objetivo proposto que era desenvolver e implementar as partes que compõem o software do sistema como também, realizar a comunicação entre as mesmas.

Para projetos futuros o grupo entende que dentre os possíveis próximos deste projeto se encontram:

- Incluir um gerador de *QR code* que permita o usuário abrir o box contendo sua encomenda;
- A implementação de um sistema de validação dos dados dos usuários via envio de foto de algum comprovante de residência;
- A implementação do módulo de recebimento de encomendas;
- A implementação de medidas de segurança no servidor, bases de dados, aplicativo e na comunicação entre os mesmos;
- A criação de um modo de acessibilidade no aplicativo, garantindo a inclusão de todos os usuários
- E a criação de uma base de dados histórica com os horários de recebimento e retirada das encomendas.

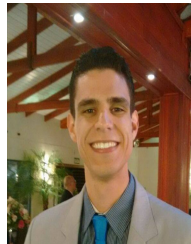
REFERÊNCIAS

- [1] B. Obodeze, *Features of a Smart Inventory Management System*. Acesso em: 01/03/2021, <https://supplychain-academy.net/smart-inventory-management-system/>
- [2] R. Smith, *An Overview of the Tesseract OCR Engine*. Acesso em: 01/03/2021, <https://github.com/tesseract-ocr/docs/blob/master/tesseractidcard2007.pdf>
- [3] C. Andaraos and M. Desimone, *Gestão de encomendas e correspondências em condomínios*. Acesso em: 06/03/2021, <https://www.sindiconet.com.br/informese/gestao-de-encomendas-e-correspondencias-em-condominios-convivencia-correspondencias>
- [4] C. E. S. Junior and D. S. Machado and V. H. dos Santos, *SmartStorageApp*. <https://github.com/dscalon/SmartStorageApp>
- [5] G1, *Entrega da correspondência é problema nos condomínios*. Acesso em: 06/03/2021, <http://g1.globo.com/bomdiabrazil/0,MUL1330819-16020,00-ENTREGA+DA+CORRESPONDENCIA+E+PROBLEMA+NOS+CONDOMINIOS.html>
- [6] K.Y Hu and T.S Chang, *An innovative automated storage and retrieval system for B2C e-commerce logistics*. Acesso em: 06/03/2021, <https://link.springer.com/content/pdf/10.1007/s00170-009-2292-4.pdf>
- [7] T. Ranzeti, *Gestão de estoque do Condomínio*. Acesso em: 07/03/2021, <https://sindicolegal.com/gestao-de-estoque-do-condominio/>
- [8] A. Calle, *Portaria virtual nos condomínios*. Acesso em: 07/03/2021, <https://calleadv.jusbrasil.com.br/artigos/381888846/portaria-virtual-nos-condominios>
- [9] L.Braga Construtora, *Entenda por que a Portaria Remota é mais vantajosa para os condomínios*. Acesso em: 05/03/2021, <https://www.lbragaconstrutora.com.br/portaria-remota/>
- [10] A.M. Becker, *Identificação de Alvo por Visão Computacional*. Acesso em: 04/03/2021, <https://repositorio.ifsc.edu.br/bitstream/handle/123456789/297/TCC%20-%20ASAPH%20MROSS%20BECKER.pdf?sequence=1&isAllowed=y.pdf>
- [11] D.W. Son and Y.S. Chang and N.U. Kim and W.R. Kim, *Design of Warehouse Control System for Automated Warehouse Environment*. Acesso em: 04/03/2021, <https://ieeexplore.ieee.org/document/7557755>
- [12] O.B.S. Filho, *Reconhecimento de Padrão de Etiquetas 2D*. Acesso em: 04/03/2021, http://repositorio.unitau.br:8080/jspui/bitstream/20.500.11874/704/1/Osias%20Baptista%20de%20Souza%20Filho_seg.pdf
- [13] B.C.S. Ribeiro and J.V.D dos Santos, *Desenvolvimento de Sistema para Controle de Estoque e Consumo de Medicamentos para Disponibilidade dos Cidadãos nos Postos UBS da Região do Centro de Guarulhos*. Acesso em: 07/03/2021, <http://revistas.ung.br/index.php/computacaoaplicada/article/view/3528/3217>
- [14] J.F.P Souza, *Projeto de um novo serviço de entrega business to consumer*. Acesso em: 07/03/2021, <https://repositorio-aberto.up.pt/bitstream/10216/93217/2/31749.pdf>
- [15] Correios, *Endereçador de Encomendas*. Acesso em: 12/03/2021, <http://www2.correios.com.br/enderecador/encomendas/default.cfm>

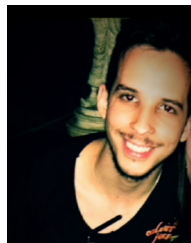


Vinícius Henrique dos Santos: Graduando em Engenharia de Controle e Automação pela UFSC - Universidade Federal de Santa Catarina, Campus Blumenau.

E-mail: vinicius.h.santos31@gmail.com



Dartagnan Scalon Machado: Graduando em Engenharia de Controle e Automação pela UFSC - Universidade Federal de Santa Catarina, Campus Blumenau. **E-mail:** dscalon@live.com



Carlos Eduardo dos Santos Junior Graduando em Engenharia de Controle e Automação pela UFSC - Universidade Federal de Santa Catarina, Campus Blumenau. **E-mail:** juninhocb@hotmail.com



Lucas William Junges: Graduando em Engenharia de Controle e Automação pela UFSC - Universidade Federal de Santa Catarina, Campus Blumenau. **E-mail:** lucaswilliamjunges@gmail.com