UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

TIBÉRIO GADELHA MAHON GOMES

HELPSUS!:
PLATAFORMA PARA AUXILIAR PROFISSIONAIS DAS UNIDADES DE PRONTO ATENDIMENTO A ACELERAR PROCEDIMENTOS BUROCRÁTICOS DOS PACIENTES.

CAMPINA GRANDE - PB

2023

TIBÉRIO GADELHA MAHON GOMES

HELPSUS!:
PLATAFORMA PARA AUXILIAR PROFISSIONAIS DAS UNIDADES DE PRONTO ATENDIMENTO A ACELERAR PROCEDIMENTOS BUROCRÁTICOS DOS PACIENTES.

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Professor Dr. João Arthur Brunet Monteiro.

CAMPINA GRANDE - PB

2023

TIBÉRIO GADELHA MAHON GOMES

HELPSUS!:
PLATAFORMA PARA AUXILIAR PROFISSIONAIS DAS UNIDADES DE PRONTO ATENDIMENTO A ACELERAR PROCEDIMENTOS BUROCRÁTICOS DOS PACIENTES.

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Professor Dr.(a.) João Arthur Brunet Monteiro

Orientador – UASC/CEEI/UFCG

Professor Dr.(a.) Robert Kelly

Examinador – UASC/CEEI/UFCG

Professor Tiago Lima Massoni

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 14 de fevereiro de 2023.

CAMPINA GRANDE - PB

RESUMO (ABSTRACT)

At the moment, the emergency care units (“Unidades de Pronto Atendimento” - UPA) of Campina Grande don’t have a system to persist data and perform operations, everything is filled in manually and stored on paper. This practice requires time, repetitive work and risk of losing data.Therefore, the present work aims to develop HelpSUS!. It’s a system that works as an electronic medical record for the UPAs. Through this system, employees will be able to register patients, manage consultations, add patient treatment data, request and manage exams and medications, issue sick-notes and view the patient’s history.

HelpSUS!: Plataforma para auxiliar profissionais das Unidades de Pronto Atendimento a acelerar procedimentos burocráticos dos pacientes.

Tibério Gadelha Mahon Gomes
Departamento de Sistemas e Computação
Universidade Federal de Campina Grande
Campina Grande, Brasil

tiberio.gomes@ccc.ufcg.edu.br

João Arthur Brunet Monteiro
Departamento de Sistemas e Computação
Universidade Federal de Campina Grande
Campina Grande, Brasil

joao.arthur@computacao.ufcg.edu.br

Resumo

Atualmente, as Unidades de Pronto Atendimento (UPA) de Campina Grande não possuem sistema para persistir os dados e realizar operações, tudo é preenchido manualmente e armazenado em papel. Esse tipo de prática demanda tempo, trabalho repetitivo e tem riscos de perda dos dados. Nesse sentido, o presente trabalho tem como objetivo desenvolver o HelpSUS!. Trata-se de um sistema que funciona como prontuário eletrônico para as UPAs. Através desse sistema os funcionários poderão cadastrar pacientes, gerenciar atendimentos, adicionar dados vitais do paciente, solicitar e gerenciar exames e medicamentos, emitir atestado e visualizar histórico do paciente.

Palavras-chave

Saúde, prontuário, SUS, atendimento.

Repositório

https://github.com/tiberiogadelha/helpsus

1 Introdução

Com o objetivo de diminuir as filas das unidades hospitalares públicas e agilizar atendimentos mais urgentes, que eram realizados em Unidades Básicas de Saúde (UBS), surge em 2002 o projeto das Unidades de Pronto Atendimento (UPAs). As UPAs prestam serviços 24 horas por dia e atendem pacientes acometidos por diversos tipos de quadro clínico, estabilizando-os e realizando diagnósticos iniciais, até a transferência para uma unidade hospitalar especializada, caso seja necessário.

Durante a pandemia do coronavírus, estas unidades de saúde foram essenciais para a população, principalmente pelo fato de estarem integradas ao Sistema Único de Saúde (SUS), ou seja, são de uso público e gratuito. Muitos atendimentos foram realizados, além dos números de costume, e estas unidades conseguiram aliviar boa parte do encargo dos hospitais. No caso de Campina Grande, Paraíba, a unidade UPA 24h Dr. Adhemar Dantas tornou-se referência para atendimento de casos relacionados à COVID-19, ao longo de boa parte da pandemia. Durante este período, a Unidade não contava com um sistema de prontuário eletrônico, sendo necessário realizar toda parte cadastral e burocrática de forma manual, usando formulários de papel.

Com o aumento brusco de atendimentos, este método utilizado para gerenciar os dados cadastrais dos pacientes e as fichas de atendimento se mostrou defasado e inviável. Muito papel passou a ser necessário, o que tornou difícil o armazenamento e integridade completa dos dados, levando a um grande risco de perda e/ou danificação, além da dificuldade para encontrar informações importantes do histórico do paciente e dentre outros problemas.

Dado esses problemas, foi visto a necessidade do desenvolvimento de um sistema de prontuário eletrônico que integrasse o máximo de setores das Unidades e fosse capaz de persistir dados, realizar operações que antes eram manuais, como cadastrar paciente, solicitar procedimentos e montar históricos para auxiliar os profissionais da saúde. Com o histórico de atendimento do paciente, contendo informações como triagens, requisições de exames e medicações e parecer médico, a tomada de decisão para algum procedimento no paciente se torna muito mais rápida e eficaz.

Para avaliar a qualidade e utilidade do HelpSUS!, foi criado um questionário do tipo *Post-Study System Usability Questionnaire* (PSSUQ), em que pessoas da área de saúde, após usarem o *software*, avaliaram as 16 afirmações [6] relacionadas à usabilidade, *interface* e utilidade, classificando cada uma das afirmações numa escala de 1 a 7, onde 1 significa que concorda

fortemente e 7, discorda fortemente. Duas pessoas de cada setor responderam ao questionário, totalizando dez participantes, após testar o sistema no *browser* do próprio computador. Cada participante recebeu suas credenciais e testou as funcionalidades que pertenciam ao seu tipo de profissão, simulando o mesmo fluxo das atividades que já realizavam.

2 PROBLEMA E SOLUÇÃO

Ao iniciar cada atendimento, uma ficha contendo os dados pessoais do paciente é preenchida à mão, sem a possibilidade de resgatar dados do enfermo, caso ele tenha sido atendido anteriormente na Unidade. Esta mesma ficha é passada adiante nos demais setores, sendo complementada com requisições de exames e/ou medicamento, parâmetros vitais do mesmo, parecer médico, atestado emitido e outras informações. O trabalho de preencher esses formulários acaba se tornando repetitivo e com risco de inconsistência, já que um paciente pode retornar diversas vezes à unidade de atendimento, gerando vários protocolos diferentes, com baixa possibilidade de recuperação dessas informações e risco de em alguma dessas fichas os dados estarem incorretos. Após alguns dias do atendimento, estas fichas são arquivadas, juntamente com várias outras fichas, em caixas de papelão, sendo levadas para uma espécie de depósito.

Nos dias de hoje, essa metodologia se mostra ineficiente, uma vez que há suscetibilidade da perda dos dados, além da dificuldade para resgatar dados de um paciente. Em caso de auditoria em atendimentos realizados há muito tempo, o processo se torna ainda mais lento e com risco das informações requisitadas não serem encontradas. Ademais, com o passar do tempo, a quantidade de fichas arquivadas vai exigir espaços físicos maiores, acarretando em mais custos e dificuldade em manter esses arquivos.

Por conta do trabalho manual, fica ainda mais difícil realizar estatísticas acerca dos tipos de atendimentos realizados na unidade. Com essas estatísticas é possível focar em itens e profissionais especializados nos atendimentos mais comuns. Sendo assim, um sistema de prontuário, com esses dados em um banco de dados, facilitaria este processo de estatística e os demais problemas.

Existem vários provedores de prontuário eletrônicos, entretanto, os *software* que foram pesquisados são focados nas metodologias de atendimento usadas em hospitais de grande/médio porte e que não possuem tanta conectividade entre os setores. Além do mais, esses provedores são do setor privado que vendem a licença para o uso desses sistemas, logo o código-fonte deles não é disponibilizado. Como as UPAs são de responsabilidade municipal é necessário realizar o processo de licitação [8], procedimento no qual empresas privadas disputam entre si para fornecer algum tipo de serviço e/ou produto para uma órgão do Estado, para conseguir usar este tipo de *software*, acarretando em alto custo para os cofres públicos e demanda tempo.

Ademais, ao encerrar o contrato da licitação, outra empresa pode ganhar o próximo processo, fazendo com que o histórico dos atendimentos seja restaurado a cada licitação, causando prejuízo à população e aos profissionais de saúde, uma vez que a população perde o seu histórico e os agentes de saúde necessitam ser treinados, já que a cada troca um *software* diferente é fornecido. Com isso, a produtividade acaba sendo reduzida, principalmente para aqueles que possuem maior dificuldade com mudanças tecnológicas. Um único sistema que persista por vários anos seria o suficiente para resolver este problema.

3 HELPSUS!

Por estar em Nuvem, o sistema poderá ser acessado de qualquer lugar, usando o computador do setor ou via *smartphone*. Por este motivo, todos os acessos e informações são protegidos, necessitando realizar login. O serviço de computação em nuvem escolhido para o projeto foi a Amazon Web Service (AWS), usando instâncias dos produtos *Elastic Compute Cloud* (também conhecido como EC2, provedor responsável pelo servidor) e o *Relational Database Service* (ou RDS, serviço voltado para banco de dados). Para separar as permissões, o sistema conta com dois tipos de usuários diferentes: o funcionário e o administrador. O administrador consegue realizar as mesmas operações que um funcionário, com a diferença de poder aceitar ou rejeitar um usuário cadastrado. Já as permissões de um funcionário variam de acordo com o setor que o mesmo está alocado.

No primeiro acesso, o funcionário irá se registrar, inserindo vários dados pessoais, a função e o número do conselho de classe profissional [3]. Feito isso, será necessário que o usuário aguarde aprovação de um administrador para poder ter acesso direto ao sistema.

As funções do sistema foram divididas por setor, assim como é em uma UPA, onde cada um possui funções bem específicas e definidas. Cinco setores foram criados no HelpSUS!, que são eles: recepção, triagem, consultório, farmácia e laboratório. A figura [1] apresenta o fluxo a que cada atendimento é submetido, o mesmo que o projeto implementa, e uma breve explicação da função de cada setor.

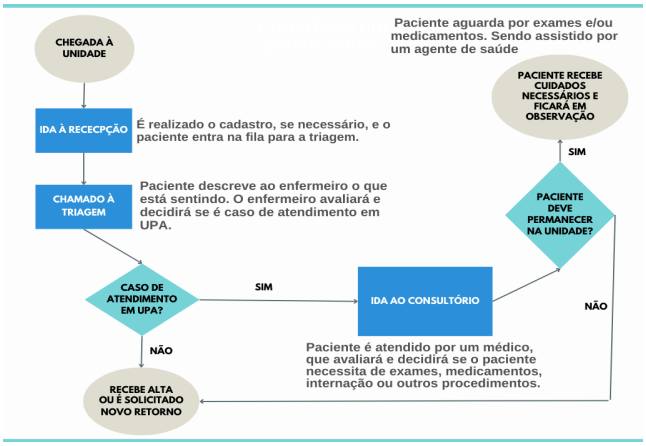

Figura **1 -** Fluxo do atendimento em uma UPA

3.1 Recepção

É neste setor que há o primeiro contato do paciente com a unidade. Os Funcionários deste setor são os recepcionistas e eles podem cadastrar e/ou editar dados pessoais de um paciente, além de adicionar ou cancelar um paciente na fila para triagem. Após o

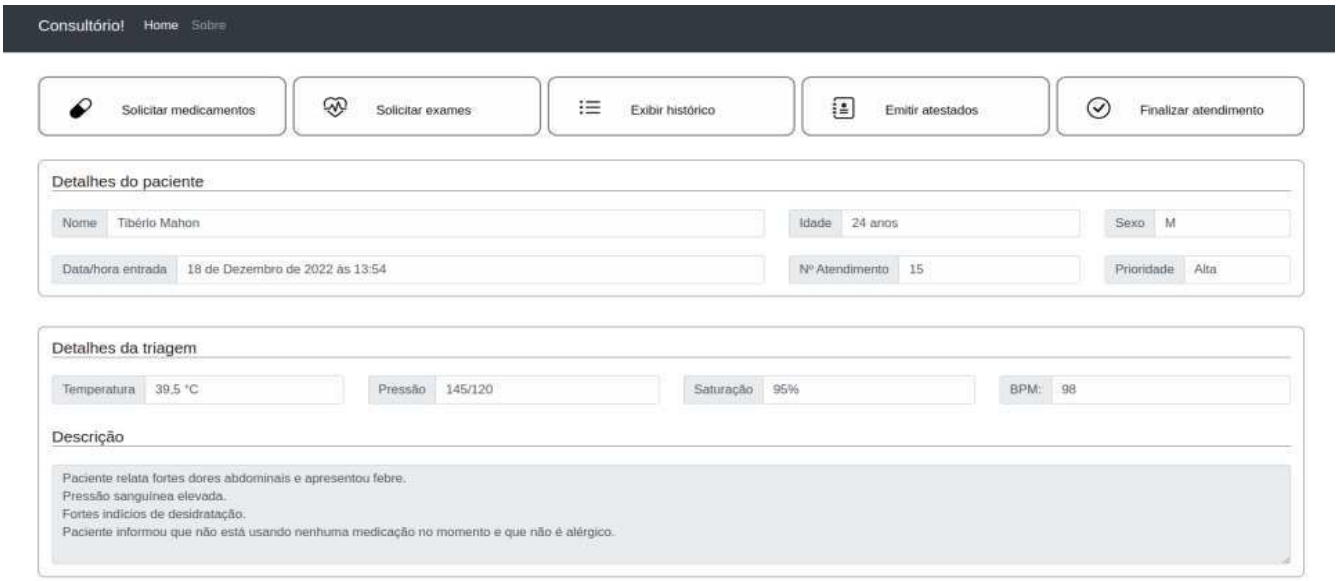

Figura 6 - Tela de um atendimento fictício no setor consultório do HelpSUS!

cadastro, o paciente aguarda ser chamado pelo setor de triagem na recepção.

Cadastro de paciente

Na primeira entrada na unidade, é necessário que seja feito o cadastro do paciente no sistema. O recepcionista solicitará dados como nome completo, data de nascimento, endereço completo e o número do CNS (o Cartão Nacional de Saúde é o documento de identificação do usuário para o SUS). O cadastro é o ponto base do sistema, pois todas as outras entidades e informações possuem referência a ele. Por se tratar de dados sensíveis, informações sigilosas serão mascaradas e só poderão ser visualizadas por quem possuir permissão.

Inserir paciente à fila

Após o cadastro, o recepcionista pode iniciar o processo de atendimento daquele paciente. Ao selecionar e confirmar o paciente que deseja atendimento, um número de protocolo é gerado e o mesmo deve ser informado ao paciente. Será através deste número que ele será chamado à triagem.

Remover paciente da fila

Em alguns casos, o enfermo desiste do atendimento e o recepcionista precisa removê-lo da fila. Logo ao confirmar o cancelamento, a solicitação de atendimento é cancelada e o próximo da fila assume a posição do desistente.

3.2 Triagem

Quem trabalha neste setor são os enfermeiros. Eles chamam os pacientes, que foram cadastrados pela recepção, para serem atendidos por ordem de chegada. O sistema dará a possibilidade destes profissionais inserirem dados vitais do paciente, como também poderão dar alta ao enfermo ou continuidade ao atendimento. A triagem é a seleção dos atendimentos que condizem com o que é proposto por uma UPA: urgência e emergência.

Visualizar e selecionar atendimentos

O HelpSUS! permitirá que enfermeiros alocados na triagem vejam os pacientes que estão aguardando serem triados, em ordem de chegada. Dados básicos do paciente serão exibidos, juntamente com o horário que foi dado a entrada na UPA. Confirmando o atendimento, o paciente será chamado e o processo de triagem iniciará.

Realização da triagem

Feita a inicialização do processo de triagem, o sistema contará com um formulário dos dados vitais do paciente, como temperatura corporal, batimentos cardíacos, saturação de oxigênio, pressão sanguínea e dentre outros que auxiliarão o médico no momento da consulta. Após preencher este formulário, o enfermeiro deve confirmar a continuação do atendimento, se o problema do paciente se enquadrar nos parâmetros de uma UPA, e informar o nível de prioridade para aquele atendimento, considerando o protocolo de Manchester [4] - a ordem de preferência para o atendimento pode ser classificada como vermelha, amarela, verde ou azul, em ordem crescente de tempo máximo de espera, na qual a vermelha representa prioridade máxima e atendimento imediato. Entretanto, se a situação do paciente não corresponder aos tipos de atendimento de uma UPA, é solicitado um novo retorno ou é feito o encaminhamento para a unidade apropriada, encerrando o atendimento.

3.3 Consultório

Local onde os médicos atuam e aguardam os pacientes que passaram pela triagem. O HelpSUS! permitirá que os médicos chamem os pacientes de acordo com a precedência de cada um. Após iniciar o atendimento, o médico terá a possibilidade de visualizar o histórico de atendimentos do enfermo, que possui resultados de triagem, solicitações de exames e medicações,

atestados e outras informações importantes. Além de visualizar, o clínico pode acrescentar solicitações, que também serão anexadas ao histórico, e finalizar o atendimento.

Visualizar e selecionar atendimentos

Assim como para a triagem, o sistema possibilitará que médicos visualizem a fila de atendimentos, ordenada de acordo com a prioridade selecionada na triagem, onde os pacientes classificados como vermelho ficarão no início da fila, respeitando os limites de espera dos outros pacientes. Ao selecionar um paciente, o processo de atendimento é iniciado e o paciente é levado ao consultório. A partir desse momento, o sistema permitirá que o médico realize diversas operações, tais como visualizar histórico, solicitar exames e/ou medicamentos e emitir atestados. Exibida na figura [6], que exemplifica um atendimento selecionado, esta funcionalidade é uma das mais complexas e com mais recursos, já que a partir dela é possível usar todas as funções disponíveis para o setor consultório.

Visualizar histórico do paciente

Um dos grandes problemas que existe hoje nas UPAs de Campina Grande é a falta do histórico do paciente, dado que as fichas são manuais e ficam arquivadas em caixas. Com base neste problema, o HelpSUS! armazenará todos os atendimentos daquele paciente, para que no momento do atendimento o clínico possa visualizar detalhes de outros atendimentos, podendo auxiliar o médico em sua conduta. O histórico é extremamente essencial, pois através dele pode ser explicado o motivo pelo qual o paciente buscou o atendimento, como também informar a evolução do quadro clínico e agilizar a conduta médica, graças aos dados acerca do paciente que serão gravados e exibidos.

Solicitar exames

Procedimento comum em consultas, os exames são ótimos indicadores que podem explicar o estado clínico de um paciente. Caso seja necessário, o clínico pode requisitar ao laboratório da unidade os exames necessários, selecionando-os na lista de exames disponíveis. Após iniciar a solicitação, a mesma já entrará no histórico do paciente e será encaminhada ao sistema do setor do laboratório.

Solicitar medicamentos

Os medicamentos são a base para o tratamento de quase todos os problemas de saúde. Por este motivo, o HelpSUS! disponibilizará um formulário para o clínico redigir um receituário, que pode ser enviado à farmácia da unidade, de forma eletrônica, ou impresso e entregue ao paciente. Assim como na solicitação de exames, a solicitação é gravada no histórico.

Emitir atestado

Outra funcionalidade essencial é a emissão de atestado. De acordo com os critérios do clínico, ele pode ou não emitir um atestado para o atendimento. Se necessário, o médico preencherá um formulário, informando a data de vencimento daquele atestado e, caso o paciente permita, a Classificação Estatística Internacional de Doenças e Problemas Relacionados com a Saúde (CID), código que classifica o problema de saúde do paciente. Os dados pessoais do paciente serão preenchidos automaticamente, sendo trazidos do cadastro do mesmo.

Finalizar atendimento

Ao término da consulta, o médico responsável deve finalizar o processo, fazendo com que o sistema marque o atendimento como finalizado, registrando o médico responsável pelo protocolo e contabilizando o tempo total do paciente na unidade. Ao realizar essa ação, o sistema considera que o paciente recebeu alta e encerra o fluxo de atendimento.

3.4 Farmácia

De responsabilidade dos farmacêuticos, aqui eles lidam com as solicitações de medicamentos, realizadas por médicos. Caso a unidade tenha as medicações solicitadas, o farmacêutico confirma a solicitação e entrega o que foi solicitado ao paciente.

Tratamento das requisições de medicamentos

As solicitações de medicamentos são encaminhadas à farmácia. No momento em que a medicação estiver pronta, o funcionário da farmácia confirma a liberação da solicitação e entrega o que foi solicitado ao paciente.

3.5 Laboratório

Os colaboradores deste setor são responsáveis pelas solicitações de exames. O setor pode cadastrar os exames disponíveis, confirmar ou rejeitar requisições. Neste departamento são realizados os exames laboratoriais e é de responsabilidade dos bioquímicos e biomédicos.

Cadastrar ou desativar exames

Com o objetivo de facilitar a criação de uma requisição de exames, os funcionários deste setor cadastram o nome e descrição de todos os exames que podem ser realizados. Entretanto, caso algum exame fique indisponível por algum fator externo, como a falta de material, o mesmo pode ficar desabilitado, sem poder ser solicitado pelos médicos.

Tratamento das requisições de exames

Todas as solicitações de exames são encaminhadas ao laboratório. Os funcionários confirmam o recebimento e iniciam a realização dos exames, após a chegada dos fluídos corporais necessários para os exames, como sangue e urina. No momento que os exames ficarem prontos, o médico será notificado e receberá os resultados dos mesmos.

4 Arquitetura

O sistema é composto por dois componentes que se comunicam entre si. O banco de dados, onde estão armazenados todos os dados necessários para o HelpSUS! funcionar; e o servidor, que

lida com o *server-side*, parte do sistema que trata das lógicas de negócio e se comunica com o banco de dados (também conhecido como *back-end*), e o *client-side*, que consiste no *front-end*, parte executada no navegador do usuário, cujas funções são de permitir que o usuário visualize e interaja com os dados, tabelas, imagens e formulários e realize as requisições ao servidor, de acordo com o que o usuário solicite. O banco de dados e o servidor rodam em máquinas diferentes, para poder permitir a escalabilidade do sistema, e ambos estão alocados em servidores na Nuvem, com alta disponibilidade, para que todos os setores das UPAs tenham acesso rápido, direto e seguro.

Todo o código do sistema foi desenvolvido em um único projeto e está disponível publicamente no GitHub.

4.1 Padrão de projeto

Tendo como objetivo facilitar o desenvolvimento e reduzir custos com servidores, foi escolhido o padrão mais comum para Django: *Model-View-Template* (MVT). No *model* fica o mapeamento das entidades que o banco de dados possuirá, enquanto no *template* é onde se encontram as partes estáticas das páginas que serão renderizadas para o usuário, o que inclui HTML, CSS, JS e as imagens usadas no sistema. E por fim, a *view* é a responsável pela implementação das regras de negócio, conexão com os modelos e retornar os templates requeridos.

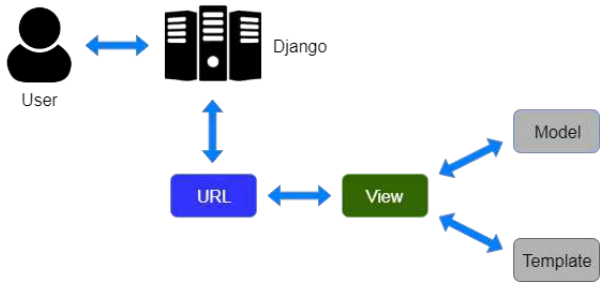

Figura 2 - Arquitetura MVT [5]

A figura [2] ilustra o fluxo que uma solicitação percorre em um servidor que usa arquitetura MVT, onde o usuário realiza uma requisição ao servidor, que buscará com base na URL a *view* responsável pela qual o usuário está solicitando. Esta *view* pode realizar ações no banco de dados, como consultas, inserções, atualizações ou remoções, e/ou retornar a página correta para o contexto requisitado, ou seja, retornar o *template*. Antes da *view* prosseguir para as demais etapas, ela pode possuir recursos para validar o usuário requisitante, verificando o seu login e suas respectivas permissões, devolvendo uma página padrão que informa a falta de permissão, nos casos em que houver tentativa de fraude nas requisições ou o usuário tentar acessar algo que não possui permissão. Este tipo de recurso é conhecido como *middleware* e foi amplamente utilizado no HelpSUS!, principalmente para validar o usuário que estava logado, suas permissões e o endereço IP do mesmo, evitando que dispositivos sem autorização não consigam operar no sistema. Após a *view* terminar suas ações, a resposta é devolvida ao cliente requisitante.

A *view* só solicita a parte do *model* quando há necessidade de executar consultas, inserções, atualizações ou remoções no banco de dados. Caso contrário, apenas as partes da *view* e *templates* são utilizadas. No caso do projeto desenvolvido, toda *view* utiliza de algum *template*, isso significa que toda requisição, após realizar as operações necessárias, retorna como resposta uma página para o usuário visualizar.

Além do fato de ser o padrão mais popular para Django, um grande fator para a escolha foi a relação custo-benefício em termos de gastos com o servidor. Diferentemente de outros *frameworks* e padrões que necessitam de ao menos duas instâncias de servidor, em que uma é dedicada ao *back-end* e outra para o *front-end*, o HelpSUS! usando o MVT se mostrou suficiente com apenas uma instância básica, isso porque o projeto não realiza operações que exigem grande poder computacional e também não terá inúmeras requisições simultâneas, considerando o cenário atual das UPAs.

4.2 Backend e Frontend (servidor)

O servidor é o responsável por responder às requisições dos usuários, através de requisições feitas pelo front-end. Cada chamada ao servidor é respondida com uma página que contém os dados solicitados, incluindo imagens, textos, tabelas e outras informações.

No lado servidor, o sistema usa o framework Django. Django [9] é escrito em Python e é voltado para o desenvolvimento web ágil. A escolha se deu devido à completude, segurança e escalabilidade do framework e também por conta da familiaridade com Python. Outrossim, o *framework* está disponível desde o ano de 2005 e é usado por diversas grandes empresas como Spotify, Instagram, Udemy e Mozilla, passando segurança, robustez e escalabilidade, itens que são essenciais para o HelpSUS!

O *framework* conta com diversas funcionalidades já disponíveis para uso (*built-in*), que agilizam o desenvolvimento deste projeto, tais como o sistema de autenticação, gerenciamento de permissões de usuários, encriptação de senha, lida com problemas de segurança, como o *SQL injection*, faz a geração automática das informações (*migrations*) necessárias para configurar e atualizar o banco de dados e dentre outras.

Além da tratativa do *server-side*, o *framework* também lida com o retorno de templates que serão renderizados no lado do cliente. Django é capaz de retornar páginas dinâmicas e estáticas, de acordo com o contexto. Usando o padrão MVT, o Django acaba se tornando um framework híbrido, em um único servidor é possível ter o *front-end* (*client-side*) e o *back-end* (*server-side*), de forma organizada, funcional e escalável.

Django possibilita a criação de aplicativos (app) para separar as regras de negócio. Cada setor é representado por um app do Django, que também é um diretório e possui a mesma estrutura da pasta 'recepcao', exibida na figura [3]. No arquivo models.py fica a estrutura das entidades daquele app. O view.py é onde ficam as lógicas do sistema, juntamente com o service.py, que complementa a view. Toda requisição passa por o urls.py, local que ficam registradas as urls e a sua respectiva view responsável. Na pasta 'templates' ficam os templates em HTML que serão retornados ao usuário, de acordo com a view responsável, juntamente com os arquivos estáticos (CSS, arquivos em

JavaScript e imagens) que ficam na pasta ‘static’. As entidades, lógicas e funções de uso comum ficam na pasta ‘core’.

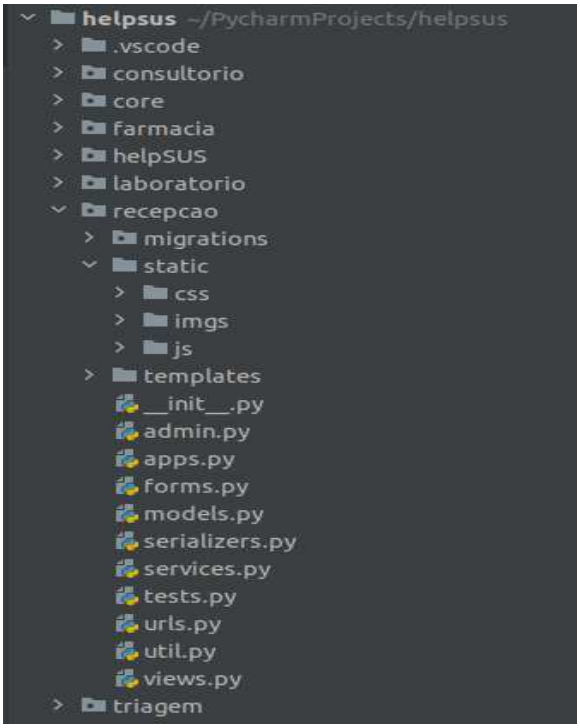

Figura 3 - Organização do projeto HelpSUS!

4.3 Banco de dados

Como o projeto possui um escopo bem definido e estruturado, sem necessidade de atividades complexas e de alto custo computacional no banco de dados, foi escolhido um banco de dados relacional (SQL). Além disso, é mais fácil de criar cópias (instâncias) do banco de dados e tornar o processamento mais fluido, caso o sistema escale bastante. Por ter mais familiaridade, ser bastante popular e *open-source,* o SQL escolhido foi o PostgreSQL.

Na secção dos modelos, exibida na figura [3], estão os modelos de objetos que são abstrações das tabelas do banco. A figura [4] exibe como as entidades se relacionam no sistema. Cada uma das entidades possuem atributos que guardam informações acerca do contexto da mesma. Para evitar repetições de atributos, foi criada uma entidade base, que possui todos os atributos comuns às demais, tais como o identificador único (ID ou chave primária) e momento da criação, atualização e remoção da entidade, e as demais classes herdam estes atributos através do princípio de herança, onde as classes filhas possuem todos os campos que a classe base possui, adicionando apenas as propriedades exclusivas da classe filha, permitindo redução de código duplicado e facilitando na manutenção.

A entidade ‘Atendimento’ é a base de várias outras, visto que é necessário haver um atendimento para que possa ser requisitado exames, medicação, triagem e etc. Os dados pessoais de cada paciente ficam em ‘Paciente’. Por ser necessário cadastrar cada paciente apenas uma única vez, é possível rastrear todas as entidades relacionadas a ele, montando de forma fácil um histórico sobre o mesmo. As entidades como ‘Triagem’, ‘Consulta’ e as requisições sempre possuem um funcionário responsável, que é representado por quem iniciou e finalizou aquela etapa do atendimento.

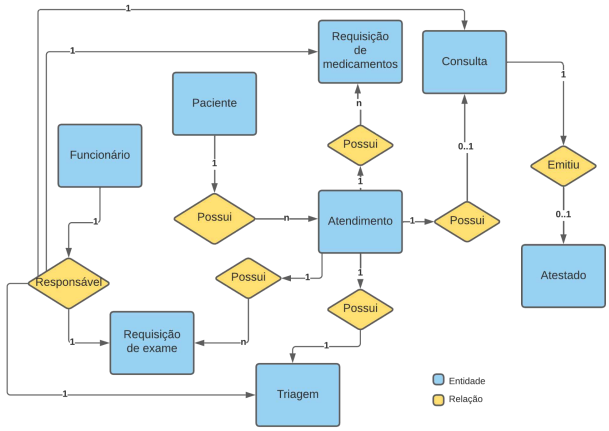

Figura 4 - Diagrama do relacionamento das entidades

Para manter a integridade dos dados e evitar problemas de concorrência no banco de dados (BD), como na atualização de um atendimento, foi usado o módulo *Transaction* do Django. O *Transaction* permite usar facilmente o princípio de atomicidade em cada requisição, fazendo com que, em caso de erro, todas as mudanças realizadas no banco de dados sejam desfeitas. Essa *feature* foi útil para impedir que alterações e criações das entidades fossem salvas, se houver algum erro durante a requisição.

Outro recurso bastante utilizado foi o *SELECT FOR UPDATE*, que permite lidar com a concorrência no banco, tornando as atualizações dos dados mais seguras, pois ele bloqueia alterações que possam ocorrer simultaneamente, criando uma fila ao nível de BD, em que apenas um requisitante acessa e/ou atualiza a informação por vez.

5 Resultados

Para avaliar a usabilidade do sistema, foi selecionado o questionário PSSUQ (Post-Study System Usability Questionnaire), seguindo o padrão de outros trabalhos semelhantes. O PSSUQ foi desenvolvido pela empresa IBM, e avalia a usabilidade, facilidade e design, com um questionário de 16 perguntas objetivas, em que para cada pergunta é dada uma resposta que varia de 1 a 7, onde 1 significa que concorda fortemente e 7, discorda fortemente. As perguntas do questionário [6] foram traduzidas e adaptadas do questionário original [7] do PSSUQ da própria IBM.

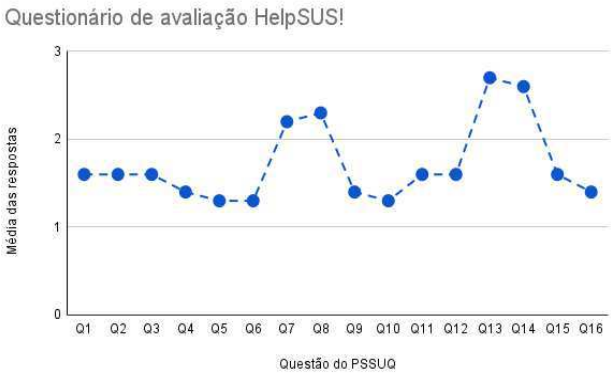

Figura 5 - Média das respostas do questionário

Foram selecionadas dez pessoas que trabalham na área da saúde, sendo duas pessoas que desempenham funções de cada setor do HelpSUS!. Após a entrega das credenciais, o entrevistador não recebe instruções prévias de como funciona o sistema, para melhorar a avaliação da usabilidade do sistema, já que se um sistema for intuitivo, o usuário consegue achar e usar as funcionalidades facilmente. Foi criado um roteiro, em que era dito o que o usuário precisava fazer após cada passo dado, de forma breve. O teste consistia em realizar o fluxo completo de um atendimento, com dados e situações fictícias. Nos casos em que houve dúvidas, como o teste foi feito presencialmente, elas foram facilmente sanadas.

Por o sistema simular a atual metodologia de atendimento de uma UPA, não houve dúvidas acerca do fluxo, apenas problemas ao preencher incorretamente algum formulário ou ao finalizar um atendimento. Com exceção do setor consultório, cada teste completo durou cerca de vinte minutos. Por o consultório ser o que mais possui funcionalidades, durou cerca de trinta minutos.

Ao fim de cada teste, o usuário foi submetido ao questionário PSSUQ na plataforma Google Forms. Por conta das perguntas curtas e respostas objetivas, não foi necessário mais que dez minutos para responder o questionário.

Após os dez participantes finalizarem os testes e avaliações, foram calculadas as médias aritméticas de cada questão e a média geral para o sistema, usando a planilha do Google Sheets, que também gerou o gráfico da figura [5]. O gráfico mostra a média das respostas em cada item do formulário, em que cada resposta segue uma escala de 1 a 7, onde o participante era livre para responder qualquer valor neste intervalo.

A média de todas as respostas do formulário foi de 1,71. Sendo assim, considerando que uma avaliação intermediária teria como média 3,5, o HelpSUS! está bem abaixo de uma classificação ruim. A maioria dos itens obtiveram médias semelhantes, podendo ser observado na figura [5], com alta aceitação e poucos itens divergiram dos demais, mas não ao ponto de classificar o sistema como ruim. O que pode justificar esta alta concordância com o sistema, é o fato do escopo do projeto ter sido montado dentro de uma própria unidade de pronto atendimento, fazendo conhecer todas as necessidades, dificuldades e fluxos de funcionamento.

As afirmativas 7, 8, 13 e 14 foram as que obtiveram maior média e estão relacionadas à satisfação com a interface e às mensagens de erro. O item Q13 “A interface do sistema estava agradável” obteve 2,7 de média e foi a maior média do questionário. Era esperado que isso ocorresse, pois não havia experiência alguma no desenvolvimento de front-end e prototipação de tela, o que levou a usar um design mais simples, sem componentes rebuscados e com cores neutras, acarretando numa avaliação com notas mais altas. Mesmo assim, a média ficou bem abaixo do que seria uma nota mediana e não impactou tanto na qualidade do sistema.

Em contrapartida, os itens que avaliam satisfação com o sistema, facilidade para usar, funcionalidades e aumento de produtividade tiveram médias baixas, que indicam alta aceitação. Em especial os itens Q5 (“Foi fácil de aprender a usar o sistema.”) e Q6 (“Acredito que eu me tornaria produtivo rapidamente usando o sistema.”) que obtiveram média de 1,3, estando muito próximos da concordância máxima. Estes itens têm fator importante na avaliação deste projeto, uma vez que a proposta principal do HelpSUS! é a de facilitar o trabalho dos profissionais de saúde, tornando-os mais produtivos.

Além da avaliação usando o questionário PSSUQ, foram feitos testes de performance para verificar o desempenho do sistema no tipo de máquina escolhida. O teste consistiu em realizar múltiplas requisições simultâneas e avaliar o uso de CPU das máquinas, até o máximo de 1000 requisições simultâneas, onde as instâncias escolhidas se mantiveram abaixo dos 85% de uso. O número máximo de requisições ao mesmo tempo é bem acima do estimado para uma Unidade.

Para os testes foram usadas máquinas da *Amazon Web Service* (AWS) para hospedar o servidor e o banco de dados. O servidor usou uma instância do tipo **t3.medium** do produto EC2 da AWS - serviço de aluguel de máquinas virtuais que podem ser configuradas conforme a necessidade - e, de acordo com a cotação gerada pela AWS *Pricing Calculator* [10], tem custo mensal de 17,93 dólares, caso seja selecionado o plano de reserva da máquina por três anos. E para o banco de dados foi selecionada uma instância do produto AWS *Relational Database Service* (RDS) - serviço de aluguel de máquinas virtuais voltadas para banco de dados - do tipo **db.tg4.micro** que tem custo mensal de 34,70 dólares, de acordo com a mesma calculadora usada anteriormente e considerando o plano de reserva da máquina por três anos. O custo do servidor do banco de dados é mais alto que o da aplicação porque máquinas voltadas para banco de dados são mais caras que as comuns, além do mais, a instância escolhida na AWS RDS conta com sistema de *backup* diário, para recuperação da base de dados em caso de algum problema. Os detalhes completos das cotações realizadas estão em anexo [11] e foram realizadas em 22 de janeiro de 2023, podendo sofrer alterações com base nas políticas da *Amazon Web Service*.

Portanto, com base nos *feedback* e resultados da avaliação, é possível concluir que o HelpSUS! é suficiente para cumprir com sua proposta: uma plataforma para auxiliar profissionais da saúde a acelerar processos burocráticos. A eficiência e concordância com o que o projeto propõe podem ser validados nos resultados do formulário, uma vez que média geral de 1,71 é considerada alta aceitação, para os parâmetros do PSSUQ. O baixo custo dos servidores também é um diferencial, mesmo usando serviços de

backup e redundância para torná-los mais seguros, a soma dos custos dos dois servidores foi de 52,63 dólares mensais.

6 Experiência e lições aprendidas

O planejamento e desenvolvimento deste projeto foi de grande aprendizado e agregação profissional, acadêmica e intrapessoal, porque foi necessário sair da zona de conforto e buscar conhecimentos em outras áreas. Por estar habituado apenas com o desenvolvimento de back-end, a produção do HelpSUS! permitiu abranger para a coleta de requisitos, gestão de projeto, desenvolvimento e prototipação de *front-end*, planejamento da arquitetura de um back-end, esquematização de um banco de dados e avaliação de produto.

6.1 Processo de desenvolvimento

A escolha do tema se deu após trabalhar durante um ano e meio na UPA Dr. Maia, em Campina Grande. Ao longo deste período, foi visto os problemas já citados anteriormente e surgiu a ideia da implementação de um sistema para resolvê-los. Foi avaliado o que seria necessário para o projeto ser aprovado e implementado para então iniciar o processo da montagem dos requisitos.

Para coletar os requisitos, foi feita uma simulação de atendimento, em que todos passos ilustrados na figura [1] foram realizados. Em cada etapa, era perguntado ao responsável como funcionava o setor, quais eram as ações que ele realizava para concluir aquela etapa e o que seria necessário para melhorar o fluxo. Após esta coleta, um *brainstorming* com um dos supervisores da unidade UPA Dr. Maia foi realizado, no qual foram discutidos os requisitos coletados e o que poderia ser acrescentado. Com isso, os requisitos funcionais do HelpSUS! foram montados.

Com os requisitos em mãos, um *product backlog* foi montado na plataforma Trello, onde cada requisito estava detalhado, com descrição, nível de prioridade e estimativa de tempo para a realização. Com base no *backlog*, foram estimadas cinco ciclos (*sprints*) de entregas, de quinze dias cada uma, para o total desenvolvimento do sistema. Após cada *sprint,* era realizada uma bateria de testes, para verificar se todos os requisitos da entrega foram realizados e buscar possíveis erros.

As funcionalidades com maior prioridade foram colocadas nas primeiras *sprints*, pois várias outras tinham dependência direta delas. Como exemplo de funcionalidade de alta prioridade, pode-se citar o desenvolvimento da recepção, setor que realizava o cadastro do paciente - para todas as demais funcionalidades era necessário o cadastro do mesmo.

As tecnologias usadas foram escolhidas de acordo com a familiaridade. Como o sistema é exclusivamente voltado ao usuário desktop, os gerenciadores de template do Django se mostraram suficientes. Além disso, Django é um ótimo framework para desenvolvimento de backend. Sendo assim, foi decidido que Python aliado ao Django seria uma boa opção.

Já para o provedor de serviços de computação em nuvem foi escolhida a *Amazon Web Service* (AWS) por conta da experiência prévia e a alta quantidade de materiais disponíveis na internet, que auxiliaram na utilização dos produtos EC2 e RDS. Apesar de existir outros provedores mais baratos, levaria bastante tempo para aprender e poderia atrasar o cronograma do projeto.

6.2 Desafios

Sobre o desenvolvimento do HelpSUS!, o maior desafio foi a prototipação e desenvolvimento das telas, pois não havia nenhuma experiência nesta área. Para mitigar os riscos, foi escolhido um modelo de interface mais simples e sem muitos detalhes, focado na funcionalidade. Além disso, o uso de *Bootstrap*, uma ferramenta com pacotes de estilizações prontas para diversos elementos, facilitou a lidar com responsividade da tela e o padrão do design das fontes dos textos, tipo dos botões, tabelas e dentre outros elementos.

Outro desafio foi a avaliação do sistema, já que esses profissionais costumam ser de difícil acesso e o tempo livre é bastante curto. Sendo assim, foi necessário dividir a avaliação em três dias, para não ocupar muito tempo de uma única vez. Para evitar atrapalhar o trabalho dos entrevistados, os testes foram realizados próximos ao horário do almoço, pois, segundo alguns funcionários, este é o horário de menor fluxo de atendimento, com exceção da madrugada.

Seguir o cronograma de desenvolvimento foi bastante árduo, as cinco *sprints* planejadas inicialmente não foram suficientes e foi necessário adicionar mais uma, totalizando seis ciclos de entrega, de 15 dias cada um. Foi gasto mais tempo que o esperado no desenvolvimento do *front-end* e para realizar as configurações para a aplicação rodar na AWS.

7. Trabalhos futuros

Considerando que um sistema de prontuário eletrônico é essencial para um bom funcionamento de qualquer tipo de unidade de saúde, o HelpSUS! se mostra eficiente para cumprir este papel. Por se tratar de um *software* que implementa o fluxo baseado na metodologia de atendimento das UPAs, há pouca mudança que possa ocorrer nas atuais funções do sistema, pois este mesmo fluxo já vem sendo usado há muitos anos e se mostrou eficiente. Entretanto, há alguns itens que podem ser acrescentados para tornar a usabilidade melhor e facilitar ainda mais a rotina dos funcionários.

Um ponto importante seria o desenvolvimento de testes de unidade e integração, para garantir que todos os fluxos funcionem como esperado e continuem funcionando, em caso de novas mudanças. Para isso, seria necessário a implementação de testes para cobrir ao menos 75% do projeto. Como o projeto não é tão grande ainda, esta atividade não exigiria muito tempo.

A integração do HelpSUS! com alguns outros sistemas de saúde o tornaria ainda mais útil. Um bom exemplo de integração seria com o sistema nacional para regulamentação de internações hospitalares (SISREG), para facilitar a exportação de dados entre estes dois *software*, agilizando o processo de transferência dos pacientes - as informações do paciente no SISREG são alimentadas manualmente -, já que as Unidades de Pronto Atendimento não foram idealizadas para lidar com internações e são dependentes dele para isto. Outra *feature* interessante seria a implementação de sistema de chat entre os setores, para colaborar com a comunicação instantânea.

Caso o sistema venha a ser implantado nas UPAs, o constante *feedback* das pessoas que o usarão renderá ainda mais funcionalidades. Permitindo que o sistema fique cada vez mais completo.

8 REFERÊNCIAS

[1] Unidade de Pronto Atendimento (UPA 24h): o que é, quando usar, diretrizes e competências. Disponível em: <https://www.gov.br/saude/pt-br/assuntos/saude-de-a-a-z/u/unidade-de-pronto-atendimento-upa-24h-1>. Acesso em: 27 de outubro de 2022.

[2] MG ganhará mais uma Unidade de Pronto Atendimento - 04/09/2009. Disponível em: <https://bvsms.saude.gov.br/bvs/pacsaude/not_04092009_mg.php>. Acesso em: 27 de outubro de 2022.

[3] O que é Conselho Profissional de Classe?. Disponível em: <https://simcarreira.com.br/conselho-profissional-de-classe/> Acesso em: 27 de outubro de 2022.

[4] Protocolo de Manchester: O que é e como implementar.. Disponível em: <https://www.totvs.com/blog/instituicoes-de-saude/protocolo-de-manchester> Acesso em: 20 de novembro de 2022.

[5] Imagem sem direitos autorais retirada da Internet. Disponível em: <https://www.tabnews.com.br/jackson541/tutorial-de-django-introducao>

[6] Questionário PSSUQ para avaliação do HelpSUS!: https://docs.google.com/document/d/1Nb4jFluuDY9I9mCchfK56e4FLJsQoz1GX1WfuKvLZPI/edit?usp=sharing

[7] PSSUQ. Disponível em: <https://uiuxtrend.com/pssuq-post-study-system-usability-questionnaire/>

[8] O que é licitação? Disponível em: <https://conlicitacao.com.br/o-que-e-licitacao/>

[9] Django - Vantagens e desvantagens. Disponívem em: <https://conteige.cloud/django-vantagens-e-desvantagens/>

[10] AWS Pricing Calculator. Disponível em: <https://calculator.aws/>

[11] Cotações dos servidores na AWS. Disponível em: <https://drive.google.com/file/d/1kZi-WECPVRUgalhisQWzQuFwe1s1fuPU/view>