INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Web Performance

MÁRIO RUI FERREIRA DINIS DE MESQUITA
Julho de 2020

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Web Performance

Mário Rui Ferreira Dinis de Mesquita

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Engenharia de Software

Orientador: Fernando Jorge Ferreira Duarte

Supervisor na Empresa: Francisco Ramalho

Porto, julho de 2020

Dedicatória

Aos meus pais, por toda a confiança que depositaram em mim.

Resumo

O presente documento visa descrever o processo de investigação e desenvolvimento do projeto realizado na empresa Glintt HS, no âmbito da unidade curricular Tese / Dissertação / Estágio, do Mestrado em Engenharia Informática, na área de especialização de Engenharia de Software, no Instituto Superior de Engenharia do Porto.

Este projeto tinha como objetivo desenvolver um processo de avaliação de uma aplicação da empresa e a aplicação de metodologias que permitissem aperfeiçoar o desempenho da mesma. Para isso, foram estudadas várias ferramentas que permitiam avaliar o seu estado inicial permitindo qualificá-la e reunir o máximo de informação relativamente à sua performance. No entanto, para se entender os valores obtidos no estudo do caso, foram pesquisadas e definidas métricas. Depois de agrupada toda a informação, prosseguiu-se para a análise e revisão de código com o objetivo de encontrar pontos críticos que afetassem o comportamento da aplicação. Existem metodologias próprias para lidar com os problemas mais frequentes neste tipo de aplicações, e que, quando aplicadas corretamente, apresentam resultados que podem ser essenciais na sua resolução.

Posteriormente, pretendeu-se avaliar a aplicação com as alterações a que foi submetida e verificar que se encontrava num estado de otimização elevado usando testes de performance, inquéritos e a comparação entre valores iniciais e atualizados das métricas definidas.

Palavras-chave: Performance, Atendimento (aplicação da Glintt), Otimização Web

Abstract

This document aims to describe the process of research and development of the project carried out at the Glintt HS company, within the scope of the Thesis / Dissertation / Internship, of the Master's course in Computer Engineering, in the specialization area of Software Engineering at the Polytechnic of Porto – School of Engineering (ISEP).

The objective of this project is to develop a process of evaluation and application of methods that allow to improve the performance of the company's applications. So, several tools were studied to assess the initial state the initial state of an application, allowing it to qualify and gather as much information as possible regarding its performance. However, in order to understand the values obtained, certain relevant metrics were researched and defined in the case of study. After grouping all the information and understanding of it, we proceed to the analysis and review of code in order to find critical points that affect the applications behavior. Nevertheless, there are methodologies that are associated with frequent problems within this topic and that, when applied correctly, present positive improvements and that can be essential in their resolution.

Subsequently, it is intended to evaluate the application with the changes it has undergone and verify that it is in a state of high optimization through tests, surveys and comparison between initial and updated values of metrics.

Keywords: Performance, Atendimento (Glintt's application), Web Optimization

Agradecimentos

À minha família e a todos os que me acompanharam neste caminho.

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos e Símbolos

Lista de Acrónimos

ACID	Atomicidade, Consistência, Isolamento, Durabilidade
AJAX	Asynchronous JavaScript + XML
API	Application Programming Interface
ATD	Atendimento
AWS	Amazon Web Servers
BASE	Basically Available, Soft State, Eventually Consistent
CI	Continuous Integration
CPU	Central Processing Unit
CRUD	Create, Read, Update, Delete
CSS	Cascading StyleSheets
CSV	Comma Separated Values
DNS	Domain Name System
DOM	Document Object Model
DSL	Domain Specific Language
DevOps	Development Operations
FCP	First Contentful Paint
FP	First Paint
FTP	File Transfer Protocol
GUI	Graphical User Interface
GWT	Google Web Toolkit
HDD	Hard Disk Drive
HP-UX	Hewlett Packard Unix
HTML	HyperText Markup Language

HTTP	HyperText Transfer Protocol
ISEP	Instituto Superior de Engenharia do Porto
JDBC	Java Database Connectivity
JMS	Java Message Service
JS	JavaScript
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
LDAP	Lightweight Directory Access Protocol
NCD	New Concept Development
NTLM	New Technology LAN Manager
NoSQL	Not Only SQL
PHP	Personal Home Page – HyperText Preprocessor
PSI	PageSpeed Insights
RAM	Random Access Memory
SOAP	Simple Object Access Protocol
SPA	Single Page Application
SQL	Structured Query Language
SRT	Start Render Time
SSD	Solid State Drive
TMDEI	Tese / Dissertação / Estágio do Mestrado em Engenharia Informática do ISEP
URL	Uniform Resource Locator
XML	Extensible Markup Language
YAML	Yet Another Markup Language

1 Introdução

1.1 Enquadramento

No âmbito da unidade curricular Tese / Dissertação / Estágio, do Mestrado em Engenharia Informática, na área de especialização de Engenharia de Software, do ISEP e com a proposta de estágio curricular da empresa Glintt HS, surgiu a oportunidade de desenvolver este documento que incide num problema cada vez mais comum e com um impacto em crescimento, que consiste na *Web Performance.* Está comprovado que um segundo de atraso na resposta de uma página reduz a satisfação de um cliente em 16% [1]. Existem várias evidências que demonstram que o desempenho das aplicações é motivo de preocupação e neste documento serão abordados temas como: a avaliação da eficiência dos produtos; a identificação dos problemas que causam o seu mau desempenho; a correção dos problemas identificados; e os benefícios obtidos pela correção dos problemas identificados.

1.2 Problema

A performance das soluções é cada dia mais importante e um fator decisivo na compra e venda de produtos de *software* [2]. A Glintt pretendia dotar a sua *software house* de ferramentas, processos e mecanismos que não só conseguissem garantir que as suas soluções tem elevada performance em diversos ambientes de execução, assim como dotá-las de uma resiliência a ambientes de stress.

Atualmente, a estabilidade das aplicações encontra-se demasiado fragilizada, pelo que estas dificilmente resistirão a esses ambientes, passando a ter um mau funcionamento. Essa situação pode abalar a confiança dos clientes e, provavelmente, inviabilizar acordos para novos projetos. Por esse facto, sentiu-se a necessidade de analisar o registo adotado no desenvolvimento das aplicações e encontrar aspetos a melhorar, para que se possa enfrentar estes problemas e ultrapassá-los sem dificuldade.

1.3 Objetivo

O objetivo deste trabalho passou por fazer uma avaliação da performance de uma aplicação da empresa usando ferramentas existentes no mercado que permitam avaliar o seu estado atual. Ao submeter essa aplicação a processos de teste contínuos, são identificados os pontos mais débeis e que podem ser fonte de lacunas no desempenho destas.

Pretendia-se que fosse realizada uma pesquisa para determinar quais as melhores formas de resolver os problemas relacionados com a performance da aplicação e que estas fossem aplicadas, de acordo com as necessidades apresentadas pelos resultados da primeira avaliação. Existem várias formas de resolver os problemas, nomeadamente usando bibliotecas construídas com um propósito específico, a minificação, entre outras formas que são apresentadas mais à frente no documento.

Por fim, depois de aplicar estes processos adequadamente, a aplicação é submetida a testes semelhantes aos da primeira fase, para garantir que os problemas não constituem mais uma adversidade, e espera-se verificar uma melhoria significativa no desempenho da mesma.

1.4 Análise de valor

A solução aqui desenvolvida aponta para um aperfeiçoamento de *software* existente na empresa que por alguma razão não se encontra otimizado, causando transtornos para os desenvolvedores e sobretudo para os clientes. Com a análise e otimização das aplicações, os clientes valorizarão a mudança e reforçarão a confiança em quem produz o seu *software*, uma vez que se tornam mais eficientes, podendo em certos casos ajudar no cumprimento de prazos. Para a empresa, o valor é igualmente elevado, não só vê a qualidade dos seus produtos aumentar como ganha vantagem em relação a outros competidores. Por fim, e mais importante que tudo, assegura a satisfação do cliente. Esta análise é mais aprofundada no subcapítulo "Análise de Valor" do capítulo "Contexto e Estado da arte".

1.5 Abordagem preconizada

Na Glintt, o processo de controlo de performance das aplicações não se encontrava ao nível desejado. Existe um departamento responsável pelo controlo de qualidade das aplicações que assegura que o produto só chega ao cliente quando estiver com as devidas funcionalidades implementadas corretamente. No entanto, pouco ou nada é visto a nível de performance dessas mesmas aplicações. Perante esta situação, surgiu a necessidade de investigar ferramentas existentes no mercado, processos e metodologias que permitam aprimorar esse controlo e assegurar a sua qualidade. Tendo em conta a falta de importância dada a este problema, surgiam constrangimentos que afetavam tanto quem desenvolve como quem testa as aplicações, e por vezes os próprios clientes. Podem-se enumerar alguns desses constrangimentos:

- A nível de performance existem aplicações que não atingem os resultados desejados.
- O tempo de abertura de algumas aplicações é demasiado grande.
- Existem aplicações que nunca foram sujeitas a determinados tipos de testes, como por exemplo, testes de carga, stress e análises de performance.

- A nível interno, torna-se difícil para a equipa de qualidade testar e validar desenvolvimentos e, consequentemente, o tempo de entrega dos projetos ao cliente aumenta.
- Aumentam os erros nas aplicações e tornando-as, por vezes, inacessíveis.

Depois de um período de análise e pesquisa a ferramentas e metodologias disponíveis, efetuou-se uma adaptação das mesmas a uma aplicação da empresa. Estas ferramentas, inicialmente, foram usadas para fazer uma análise exaustiva à performance dessa aplicação e determinaram-se os focos de maior debilidade. Desta forma, foi possível entender a origem do problema. Depois de encontrados esses focos, foram aplicadas as metodologias e processos que foram pesquisados e estudados na fase inicial do projeto para a sua resolução.

1.6 Estrutura

Este documento é composto por seis capítulos, sendo eles a Introdução, Contexto e Estado da Arte, Avaliação de Soluções/Abordagens Existentes, Experimentação e Avaliação e, por fim, Conclusões e Trabalho Futuro.

O primeiro capítulo divide-se em seis subcapítulos: Contexto, Problema, Objetivo, Análise de Valor, Abordagem Preconizada e Estrutura. Nos primeiros três subcapítulos pretende-se que o leitor fique com uma noção geral do problema identificado. De seguida, apresenta-se um pequeno resumo da Análise de Valor da solução bem como a sua abordagem preconizada. Por fim, inclui-se o presente subcapítulo onde é apresentada a estrutura deste documento.

No segundo capítulo aprofunda-se o contexto e o problema, para que o utilizador compreenda a dimensão e impacto que a Performance das aplicações tem, principalmente no setor financeiro das empresas. Depois, elabora-se um estudo à Análise de Valor da solução concebida. As secções seguintes correspondem ao Estado da Arte onde se faz um estudo das metodologias existentes para a correção de certos problemas relacionados com a performance na Web e se reúnem as aplicações existentes atualmente no mercado que se relacionam de alguma forma com o problema aqui representado. Por último, são apresentadas as vantagens e desvantagens dessas aplicações.

No terceiro capítulo encontra-se descrita a avaliação das soluções e abordagens existentes relacionadas com o problema da performance das aplicações. São apresentados exemplos do problema existente e as respetivas abordagens.

O quarto capítulo incide no desenho da aplicação que se submeteu ao processo de otimização, a definição da sua arquitetura e a inclusão de diagramas para melhor compreensão da sua implementação.

No quinto capítulo procede-se à experimentação. As metodologias e ferramentas apresentadas anteriormente foram aplicadas com o objetivo de garantir um desempenho otimizado da aplicação e é neste capítulo que se relata esta investigação. Depois é realizada uma análise dos

resultados obtidos e onde são evidenciados os efeitos e consequências de cada modificação, renovação e aperfeiçoamento do código da aplicação.

Por fim, no sexto e último capítulo serão apresentadas as conclusões e deduções retiradas deste estudo.

2 Contexto e Estado da arte

2.1 Detalhes de Contexto e Problema

2.1.1 Contexto

A vontade inata do ser humano de investigar e estar constantemente na vanguarda da inovação demonstra a capacidade de persistência existente em nós. Desde sempre existe a ideia de que é sempre possível fazer mais e melhor. É devido a este auto-criticismo que o ser humano aprendeu a evoluir na criação de novas ferramentas, objetos e ideias que poderiam, à primeira vista, parecer inviáveis, e que com espírito crítico, rigor e vontade, tem vindo a conseguir implementar.

Com a constante evolução das tecnologias e serviços de *software*, depressa aumentou a comercialização de novos produtos. Este enorme aumento de novos produtos, fez com surgissem cada vez menos ideias inovadoras. Com a originalidade a desvanecer-se, outros fatores foram postos em cima da mesa. Era necessário algo que distinguisse um produto dos restantes do mesmo ramo, que tornasse a escolha dos utilizadores mais objetiva. Uma das atitudes foi o começo da valorização cada vez maior do desempenho das aplicações. Os utilizadores depositam mais confiança e preferem visitar páginas cujo desempenho é superior [3]. Existem casos comprovados onde a melhoria de desempenho apresentou um aumento significativo nas visitas e inscrições em páginas [4], e onde o baixo desempenho afetou negativamente alguns negócios [5].

2.1.2 Problema

Este documento incide sobre performance na Web. Milhões de euros são perdidos anualmente devido a problemas de performance na Web [6]. Para grandes aplicações, uns simples milissegundos podem traduzir-se em milhares de euros de prejuízo. A Amazon, uma empresa mundialmente reconhecida pela venda e distribuição de produtos, admitiu uma perda de 1% do valor das suas vendas por cada 100 milissegundos extra no carregamento da sua página. Estes valores são enormes e podem chegar aos 1.6 mil milhões de euros anualmente, apenas com o aumento de 1 segundo [6]. A Google, multinacional americana de tecnologia especializada em serviços e produtos relacionados com a internet e o motor de busca mais utilizado no mundo [7], afirmou que com o aumento de apenas 4 décimas de segundo nos resultados das procuras, perderia aproximadamente 8 milhões de euros por dia, que resultava em milhões de anúncios online não veiculados [6]. Estes exemplos, apesar de reais, são impensáveis para empresas de menor dimensão, sendo, no entanto, essenciais na consciencialização deste problema.

O dinheiro perdido é de facto um grande problema, mas uma das razões para isso acontecer, reside na perda de utilizadores. Por este facto, o principal foco das empresas deveria ser a construção de sites e/ou aplicações amigáveis e de fácil utilização, de forma, a que os utilizadores tenham a melhor experiência possível. A estratégia deve passar por primeiro atrair o maior número de utilizadores, e posteriormente, retê-los durante o máximo tempo possível. Se não houver garantia que o desempenho esteja otimizado, muito provavelmente, o site verá os seus utilizadores a visitar outras fontes (sites). Para a avaliação da qualidade de um site, entre outras, é usada a sua taxa de rejeição, que corresponde à percentagem de utilizadores que navegam para outros sites após visitarem apenas uma página. Estima-se que um site que demore 5 segundos a carregar tenha o dobro da taxa de rejeição de um site que demore apenas 1 segundo [8].

Por tudo o que foi aqui referido, é evidente a necessidade de garantir o desempenho ideal dos produtos oferecidos por uma empresa, não só para satisfação do utilizador como para o aumento de confiança do mesmo, e consequentemente, a obtenção de novos clientes.

2.2 Análise de valor

2.2.1 New Concept Development Model

O *New Concept Development Model* divide-se em três partes: o motor, os elementos chave e os fatores ambientais que influenciam os anteriores. O motor corresponde ao cerne do sistema que fortalece os componentes essenciais como a visão, estratégia, recursos, equipas e colaboração [9].

Os elementos chave deste modelo são [9]:

1. ***Oportunity identification*** – momento em que são identificadas novas oportunidades que constituam uma possível ameaça ao negócio atual, avaliando fatores económicos, tendências culturais e consumistas.
2. ***Oportunity analysis*** – momento de análise às oportunidades identificadas, onde se estuda a probabilidade de sucesso técnico e de mercado através do planeamento e da avaliação das vantagens, tendo em conta os recursos da empresa.
3. ***Idea generation & enrichment*** – momento de metamorfose, em que as oportunidades se transformam em ideias de novas tecnologias, aliadas à necessidade dos clientes e às possibilidades da empresa.
4. ***Idea selection*** – como o próprio nome indica, depois de identificadas várias ideias, terão de ser selecionadas as que vão ser desenvolvidas. O investimento nas ideias atrai responsabilidades, logo, existem critérios que terão de ser avaliados para garantir retorno, como é o caso, da definição de métodos de avaliação e análise de possíveis concorrentes.
5. ***Concept Definition*** – definição de conceito apresentada formalmente onde se evidenciam as vantagens do produto. É avaliada a viabilidade de introduzir a ideia no

processo de produção, quais os requisitos técnicos e os fatores económicos (semelhante a um planeamento de negócio).

Os fatores ambientais são responsáveis por influenciar o motor e os elementos chave. Fazem parte destes fatores, a gestão de conhecimento, as tendências de mercado, as tecnologias e a competição.

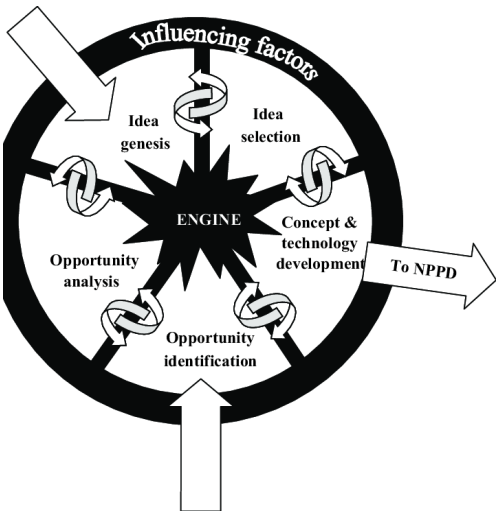


Figura 1 - New concept development model (retirado de [10])

As setas na direção interior da figura 1, relativas aos fatores de identificação de oportunidades e geração de ideias e enriquecimento, indicam os pontos iniciais para a criação de projetos. Na direção oposta (para o exterior da imagem) verifica-se o processo de Definição de Conceito que corresponderá ao último passo do produto antes de transitar para um ambiente de produção.

O projeto que aqui é relatado, integra-se da seguinte forma com os elementos chave do *New Concept Development Model*:

- ***Oportunity identification*** – Existe uma falta de preocupação no processo de controlo da performance das aplicações da Glintt, pelo que surgiu a necessidade de identificar e encontrar formas de corrigir este problema. O departamento de qualidade apenas valida se as funcionalidades de uma aplicação estão ou não implementadas corretamente, sem ter em consideração se o seu desempenho é o melhor.
- ***Oportunity analysis*** – Está comprovado pelas estatísticas apresentadas anteriormente que a melhoria no desempenho das aplicações afeta de forma positiva um conjunto de

fatores que podem, direta ou indiretamente, influenciar o aumento da receita das mesmas. Combinando esta vantagem com os problemas existentes nos produtos da empresa que necessitam de uma modificação urgente, facilmente se conclui o quão conveniente é efetuar este tipo de trabalho.

- ***Idea generation & enrichment*** – Assim que a oportunidade é definida e validada, passa-se para a caracterização de ideias. Como o objetivo é de aprimorar um produto já existente, foi feito um estudo de ferramentas capazes de avaliar a aplicação. Identificaram-se algumas capazes de fazer uma estimativa aproximada do estado da aplicação, bem como algumas que oferecem serviços de monitorização, e outras que permitem submeter a aplicação a uma carga elevada de utilizadores virtuais. Com estes resultados, é possível definir os componentes que apresentam maiores debilidades e preparar um conjunto de processos e metodologias a serem aplicadas.
- ***Idea selection*** – Feita a análise das ferramentas existentes no mercado, excluíram-se aquelas que tivessem um custo associado e as que necessitassem de alojar a aplicação nos seus servidores para se poder monitorizar a mesma. Relativamente às metodologias existentes, estas dependem da aplicação e dos problemas que esta apresenta. Por exemplo, no caso da aplicação aqui tratada (Atendimento) poucas imagens existem, pelo que utilizar metodologias como *lazy loading* ou compressão de imagens não terão efeito positivo quase nenhum.
- ***Concept definition*** – Através da análise e melhoria de performance será possível aumentar a satisfação dos clientes e possivelmente atrair outros. Isto não só é uma garantia de qualidade nos produtos, como um possível aumento nas receitas.

2.2.2 Valor

O valor de um produto está associado ao custo de fabrico e ao preço que é vendido de forma a haver lucro. Corresponde ao valor sem expectativas exteriores, tanto do comprador como do vendedor. É ainda caracterizado pela razão entre os benefícios e os respetivos custos/sacrifícios.

Avaliando os possíveis resultados do trabalho aqui descrito, encontram-se vários benefícios tanto para o cliente como para a empresa. Com a melhoria na performance das aplicações, a equipa de qualidade consegue testar e validar desenvolvimentos mais facilmente, o que naturalmente vai permitir a entrega dos projetos mais rapidamente, diminuindo os erros, permitindo tempos de resposta aceitáveis e uma maior cobertura de testes. No entanto, para se beneficiar destes aspetos existe um custo associado que a empresa terá que suportar. Entre eles pode-se enumerar o investimento em *software* de monitorização de performance, reformulação de algumas soluções já existentes e o possível investimento numa equipa responsável pela manutenção das aplicações. Espera-se que estes custos sejam recompensados com o reforço da confiança do cliente associado à sua satisfação com estas intervenções.

2.2.3 Valor para o cliente

Valor para o cliente é um conceito que pode ser interpretado de várias formas. Pode traduzir-se como o valor que um produto ou serviço tem para um cliente em comparação com as

alternativas possíveis. Este valor é positivo se o comprador considerar que obteve bons benefícios tendo em conta os custos associados.

Valor para o Cliente = Benefícios – Custos

Com o trabalho a desenvolver, apesar das aplicações não serem novas, os clientes podem sentilas como tal. Não do ponto de vista em que haverá necessidade de aprender todas as suas funcionalidades e habituar-se a uma nova aplicação, mas no sentido em que os fluxos de trabalho serão muito mais fluídos. Como está comprovado que *websites* ou aplicações lentas podem levar a um aumento do ritmo cardíaco e causar stress [11][12][13], esta melhoria na interação do utilizador com a aplicação, causará, certamente, um impacto positivo no dia a dia do cliente. Este impacto traduzir-se-á em confiança, sentimento de segurança e aumento do seu rendimento.

2.2.4 Valor percecionado

O valor percecionado é o valor que o cliente realmente pensa que o produto vale. Esta perceção é desenvolvida pelas opiniões de mercado e pelos benefícios que o cliente espera obter caso efetue a compra do produto. O modelo de valor varia de cliente para cliente pois este baseia-se nas necessidades, desejos, características e capacidades financeiras de cada um. O mesmo produto pode ter diferentes valores percecionados, uma vez que um cliente pode valorizar mais a qualidade do produto e consequentemente, pagar mais pelo serviço, enquanto que outro pode prestar mais atenção ao preço e preferir o serviço mais económico.

2.2.5 Proposta de Valor

Uma proposta de valor representa o potencial que o produto pode alcançar e onde se distingue em relação aos produtos de outros competidores, salientando os benefícios e pontos fortes de forma clara. Para o cliente entender realmente o serviço que pondera adquirir, é necessário esclarecer todas as questões pertinentes. Dependendo do produto, existem questões que surgem naturalmente e que são impreteríveis para o cliente na compreensão da finalidade do produto. A intenção da proposta de valor é de elucidar o comprador sobre como o produto vai solucionar os problemas, como se compara a outros produtos e o que o torna distinto.

O projeto aqui desenvolvido tem como objetivo resolver os atuais problemas de performance em aplicações da Glintt. Para atingir este objetivo, é necessário identificar onde se encontram os problemas mais profundos e que afetam consideravelmente o desempenho das aplicações. Posteriormente, faz-se uma análise contínua e correções às contrariedades encontradas com o intuito de as eliminar. A partir do momento em que deixam de existir adversidades, a aplicação deverá encontrar-se num estado de elevado desempenho. Depois do devido procedimento de

testes para assegurar a qualidade do trabalho efetuado, estará preparada para ser entregue ao cliente.

Para o cliente, esta operação vai trazer vantagens a vários níveis. Ao eliminar problemas de desempenho de uma aplicação que o utilizador se encontre acostumado a utilizar diariamente, causará uma sensação de confiança na empresa responsável pelo serviço, não só porque indica interesse em aperfeiçoar a experiência do utilizador, como também demonstra compromisso com a qualidade do produto. Com o aumento da satisfação do cliente para com o produto e com a redução dos seus níveis de stress (evidenciado no subcapítulo Valor para o Cliente), é provável que o seu rendimento seja influenciado positivamente.

Os clientes da empresa fazem parte de um grupo de consumidores de *software* maioritariamente relacionado com saúde (Hospitais, farmácias, etc.). A estrutura das aplicações desenvolvidas pela empresa é idêntica. Seguindo uma filosofia semelhante à anteriormente descrita, onde a aplicação em causa é estudada, são identificados os pontos de debilidade e são usados métodos adequados aos problemas encontrados, de forma a resolvê-los, o conceito de aprimorar o desempenho aqui desenvolvido, pode ser aplicado às várias aplicações da empresa.

Além de garantir a satisfação dos clientes, aumentando a sua confiança e fidelidade, as aplicações são colocadas num patamar de critério elevado e a sua qualidade será associada à empresa, cuja reputação será cada vez melhor, o que permite distanciar-se das empresas competidoras.

2.2.6 Canvas

A figura seguinte representa o modelo Canvas da solução pretendida.

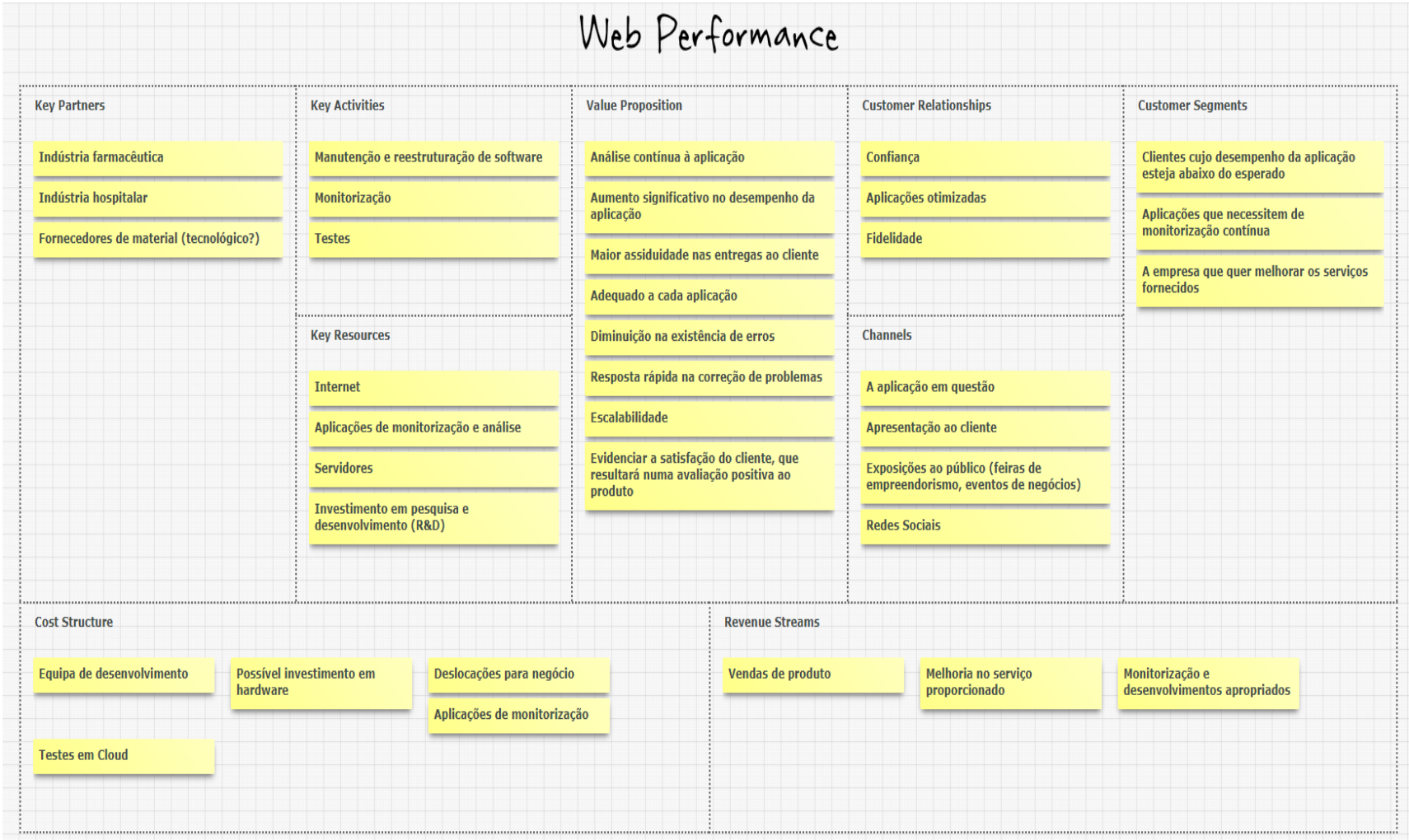


Figura 2 - Canvas

2.3 Estado da arte em soluções/abordagens existentes

O problema relacionado com a performance não é uma novidade no mundo do desenvolvimento de programas e aplicações. Com a constante luta contra a falta da performance desejada, foram encontradas algumas metodologias para tentar debelar essa situação.

Antes de apresentar essas metodologias, é necessário introduzir certos conceitos para que se possa entender melhor o funcionamento destas.

"*The perception of performance is just as effective as actual performance in many cases*" [14].

O primeiro conceito a introduzir será o de *Perceived Performance.* O desempenho percecionado refere-se ao quão rápido um utilizador pensa que uma aplicação é rápida. Não está necessariamente ligado aos valores apresentados nas estatísticas medidas por outros *softwares.* Quando o objetivo é a otimização de um *website* é o que o utilizador pensa, que vai fazer a diferença [15].

Com o objetivo de quantificar aproximadamente os valores entre os quais varia a perceção do ser humano, foram definidos e caracterizados alguns limites nos tempos de resposta de uma aplicação. [16]

- **a 0.2 segundos –** Estudos mostram que este intervalo representa o valor máximo aceitável do tempo de resposta para ser considerado instantâneo, ou seja, o intervalo de tempo a partir do qual, o utilizador se apercebe de alguma demora [17].
- **0.5 a 1 segundo –** Este é o intervalo máximo para ser considerado um comportamento imediato. Corresponde, normalmente, ao tempo de resposta entre uma conversa entre duas pessoas. A demora dentro deste intervalo de tempo é notável, mas facilmente tolerável pela maior parte dos utilizadores. No entanto, estes devem receber uma indicação muito subtil de que a sua ação está a ser processada, para que não sintam que a sua interação não foi registada. Desta forma, não se interrompe o fluxo de pensamento do utilizador. [16]
- **1 a 5 segundos** – Neste intervalo de tempo, é necessário apresentar ao utilizador algum feedback que indique que os pedidos estão a ser processados e que o sistema está a responder. Este *feedback* deverá ser sob a forma de um indicador circular simples ou uma barra de progresso (sem percentagem) pois não há necessidade de chamar a atenção do utilizador para informações adicionais. [18]
- **5 a 10 segundos** – De acordo com o *National Center for Biotechnology Information at the US National Library of Medicine*, o período de atenção médio de um ser humano baixou dos 12 segundos em 2000 para 8.25 segundos em 2015. Pode-se concluir que neste intervalo de tempo, o utilizador ainda se encontra focado na sua tarefa, mas facilmente é distraído. Esta é a fronteira limite da tolerância do utilizador e exige-se uma solução mais completa para conservar a sua atenção. Para intervalos como este, ou mais longos, é necessário apresentar informação específica sobre o que está a acontecer e envolver o utilizador no processo. Para isso, deve ser apresentado um indicador dinâmico que indique o progresso da atividade, de forma clara e explícita.

Estes valores apresentados referem-se a tempos de uma página Web, no entanto, sabe-se que a perceção do tempo no quotidiano é uma questão de contexto. Um indivíduo pode esperar 15 minutos para receber a comida num restaurante, contudo pode não esperar 2 minutos para carregar uma página Web. Isto é expectável devido à relação de retorno e investimento que a pessoa fez. São investidos 15 minutos para receber uma refeição agradável e que vai ajudar a pessoa a satisfazer as necessidades durante um longo período. Enquanto que esperar 2 minutos para uma página carregar, não compensa o seu retorno pois pode nem ser o conteúdo que o utilizador desejava e pode nem despender 15 segundos nela.

O segundo e o terceiro conceito a introduzir estão interligados e caracterizam-se por espera ativa e espera passiva. A primeira aparenta ser menos tempo de espera porque significa que a pessoa está ativamente envolvida numa ação ou momento. Consequentemente, o tempo percecionado por ela será menor do que a realidade, uma vez que esta se encontrava concentrada no que está a fazer e não se apercebe do tempo decorrido. O caso da espera passiva consiste precisamente no contrário, já que o tempo de espera será percecionado como maior do que é na realidade. A pessoa está somente focada no tempo que está a passar, não desviando a sua atenção para outros interesses. Com o aumento desta espera, vão aumentar os sintomas de impaciência, frustração e ansiedade da pessoa em questão. Pode-se concluir que, para suavizar o tempo de espera de um utilizador e melhorar a sua perceção de tempo, deve-se reduzir o tempo de espera passiva e dilatar o tempo de espera ativa.

Apresentados os conceitos fundamentais de perceção temporal, são descritas de seguida, as metodologias que ajudam a idealizar abordagens e soluções para problemas como o que é apresentado neste documento.

2.3.1 Metodologias

Na presente secção serão apresentadas certas metodologias que foram consideradas importantes no desenvolvimento do projeto. A utilização de uma metodologia não invalida as restantes, uma vez que estas podem ser aplicadas simultaneamente e podem complementar-se.

2.3.1.1 Preemptive Start

A técnica *preemptive start* [19] consiste em entrar num processo com uma fase ativa e permanecer neste estado o máximo de tempo possível antes de sujeitar o utilizador a uma espera passiva. Como referido anteriormente, o tempo de espera ativa por vezes, faz com que o utilizador nem se aperceba que está, efetivamente, à espera de algo. Consequentemente, para o cérebro humano, um início preemptivo significa a mudança do ponto de início do processo para um ponto mais perto do fim, que dará a ilusão ao utilizador que a duração do processo foi mais curta.

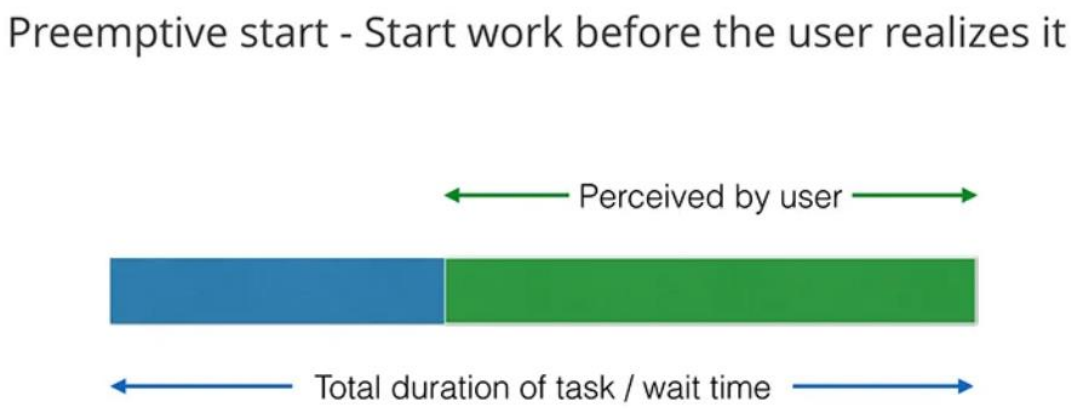


Figura 3 - Exemplo gráfico de Início Preemptivo (Retirado de [18])

Em certas situações, a fase ativa no início de um processo pode causar alguma dúvida na sua aplicação. De forma a compreender melhor esta metodologia, são descritos de seguida alguns exemplos hipotéticos do quotidiano adaptados da revista digital *SmashingMagazine,* no artigo "*Why Performance Matters*" [16].

Aeroporto

A equipa de gestão de um aeroporto recebe reclamações relativas aos longos tempos de espera para receberem a bagagem quando chegam ao aeroporto de destino. Para resolver o assunto, os responsáveis do aeroporto decidiram aumentar o número de trabalhadores no tratamento das bagagens. Esta reação permitiu que as bagagens fossem descarregadas mais rapidamente dos aviões. No entanto, o caminho que estas fazem até chegarem ao tapete de entrega aos passageiros, é maioritariamente controlado por máquinas, logo, o tempo de espera diminuiu, mas não o suficiente para que os passageiros ficassem satisfeitos, tendo-se mantido o número de reclamações. Depois de alguma recolha de informação sobre o assunto, os responsáveis chegaram à conclusão que as bagagens demoravam, em média, 8 minutos para chegar ao ponto de entrega enquanto os passageiros demoravam cerca de 1 minuto a chegar ao mesmo local. Isto significa que os passageiros esperavam aproximadamente 7 minutos para receberem as respetivas bagagens. Se atendermos aos conceitos introduzidos previamente, existe apenas 1 minuto de espera ativa e 7 minutos de espera passiva.
Utilizando a metodologia do início preemptivo, os responsáveis decidiram afastar as portas de chegada do terminal principal e redirecionaram as malas para um ponto mais distante. Ao aplicar estas mudanças, os passageiros teriam de fazer uma caminhada de cerca de 6 minutos (fase ativa) para chegar ao local de entrega das malas e esperar apenas 2 minutos (espera passiva) para que estas chegassem. Ao alterar a distância entre os dois pontos de interesse dos passageiros, foi possível beneficiar da perceção do tempo por parte dos passageiros e os trabalhadores do aeroporto trataram das bagagens da mesma forma, tendo demorado o tempo normal para o processo.

WEB

Esta metodologia pode ser usada na Web nas funções de procura, por exemplo. Assumindo que um campo de procura está presente em todas as páginas de uma aplicação, mas os resultados das procuras necessitam de passar por um processo trabalhoso e complexo de ordenação e filtragem. Pode-se então começar um pré-carregamento da informação assim que o utilizador começar a escrever no campo de procura. Isto vai antecipar que a informação já esteja praticamente pronta a ser apresentada quando o utilizador entra no painel ou página que contenha os resultados.

2.3.1.2 Conclusão Antecipada

Em oposição ao início preemptivo, que se foca em aproximar o ponto inicial de um processo mais perto do seu ponto final, a conclusão antecipada [20] tem como objetivo aproximar o ponto final de um processo do seu ponto inicial. O uso mais comum desta metodologia verifica-se nos serviços de transmissão de vídeo. Quando um utilizador pretende visualizar um vídeo, não espera que este termine de carregar totalmente. Assim que uma pequena parte esteja disponível para ver, o vídeo começa. Assim, o ponto final do processo aproxima-se do ponto inicial. O utilizador encontra-se num estado de espera ativa, pois o vídeo ainda não terminou de carregar, mas já é possível vê-lo enquanto a restante parte é carregada em segundo plano.

Esta metodologia pode ser aplicada em aplicações Web nos momentos em que uma página está a carregar. O intervalo entre o envio de um pedido até à apresentação do resultado no browser corresponde a uma espera passiva do utilizador, portanto pretende-se que este seja o mais curto possível. Uma das soluções passa por apresentar as partes essenciais da página logo que esta esteja pronta. Não há necessidade de esperar que todos os elementos da página estejam carregados, caso não afete a performance da página. Pode-se definir que a informação que não esteja imediatamente visível no início da página, seja apresentada depois dos elementos que o utilizador vai ver em primeiro lugar. Existem bons exemplos de páginas reconhecidas que utilizam este método para oferecer ao utilizador uma experiência fluída e de resposta rápida, como é o caso dos sites da Amazon e do eBay. Depois de efetuar inúmeros testes ao carregamento da página da Amazon, foi possível observar que eram necessários à volta de sete segundos para a página estar completamente carregada. No entanto, graças a aplicações como GTmetrix [21] e PageSpeed Insights [22], é possível observar que, no primeiro segundo, o topo do site já se encontra visível, ficando o utilizador numa espera ativa enquanto o resto da página é carregada, sem se aperceber. Dar prioridade ao conteúdo visível de uma página, é um bom exemplo de conclusão antecipada. Apresentar algo de essencial aos utilizadores assim que for possível, transmite a ideia de que o carregamento da página é realizado mais rapidamente.

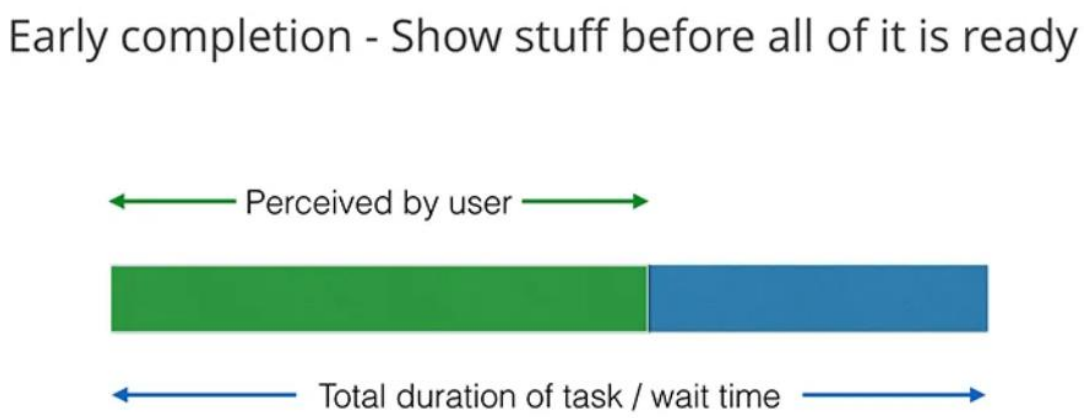


Figura 4 - Exemplo gráfico de Conclusão Antecipada (retirado de [18])

2.3.1.3 Pré-Procura, Pré-Renderização e Pré-Carregamento de conteúdo

A pré-procura [23] consiste em notificar o navegador para que este procure certos recursos antes de serem necessários. Serve como uma navegação futura do utilizador. Se uma página solicitar um pedido de pré-procura de recursos para a página seguinte, os recursos e pedidos críticos podem ser concluídos em paralelo. [24] O *browser* armazena esses recursos em cache permitindo uma entrega mais rápida dos dados assim que forem solicitados.

A pré-renderização é uma prática recente, que consiste em dar indicação ao navegador para descarregar os recursos e renderizar uma página inteira em segundo plano. Esta capacidade pode ser importantíssima quando aplicada na situação certa. Como exemplo, no caso de um utilizador se encontrar a ler os termos e condições de uma página, é muito provável que vá aceitar e terá de ser transferido para uma nova página. Enquanto este faz a leitura dos termos e condições, a segunda página já poderá estar a ser descarregada e quando o utilizador aceitar os termos e condições, a nova página é apresentada quase instantaneamente. Esta metodologia pode ser usada neste caso, pois é muito provável que o utilizador passe à página seguinte. Existem outros casos em que esta prática pode ser aplicada, como por exemplo, quando uma página consome muitos recursos e demora muito a ser carregada. No entanto, estes casos nem sempre se verificam e por vezes a rede disponível ao utilizador pode ser limitada, havendo a necessidade de utilizar a técnica com ponderação e se o seu uso for justificado.

O pré-carregamento tem uma funcionalidade semelhante à metodologia previamente descrita, com a diferença de atuar apenas em determinadas partes de uma página (em vez da página toda), como imagens ou scripts que não estão a ser utilizados, mas que poderão ser dentro em breve. O tamanho dos sites atualmente aumenta significativamente devido à abundância de imagens e *scripts* enormes, logo, se for possível obter a vantagem de descarregá-los antes de serem necessários, verificar-se-á uma melhoria nas respostas do site e na abertura de novas páginas.

2.3.1.4 Minificação

Esta metodologia corresponde ao processo de remoção de elementos desnecessários e de reescrita de código para reduzir o tamanho de um ficheiro [18].

É frequentemente usada em ficheiros HTML, CSS e JavaScript. Ao reduzir o tamanho dos ficheiros, estes são transferidos mais rapidamente e, naturalmente as páginas são carregadas mais rapidamente. Há várias formas de minificar informação, sendo as mais básicas, a remoção de comentários, espaços desnecessários e quebras de linha. Os desenvolvedores devem ter o código indentado e devidamente comentado para cumprir as regras de boa programação, no entanto para o navegador essas linhas e caracteres são ignorados. Ao utilizar esta metodologia, todas essas regras vão ser desconsideradas e podem até ser alterados os nomes sugestivos das variáveis para um único caracter. Em ambiente de desenvolvimento, o resultado da aplicação desta metodologia poderia ser desastroso pois o desenvolvedor podia deixar de conseguir compreender convenientemente o código que produziu, sendo ainda pior para outros desenvolvedores que não tinham participado na produção desse código e que teriam que mais tarde analisar e alterar esse código. Assim, a minificacão deverá acontecer somente em ambiente de produção, onde os ficheiros são separados do ambiente de desenvolvimento. Isto torna a alteração do código mais segura e permite aos desenvolvedores manter as boas práticas de programação e compreensão do seu trabalho.

2.3.1.5 Tree shaking

Esta metodologia [25] é usada para eliminar o código que a aplicação não emprega. Quando se importam e exportam módulos em JavaScript, na maior parte dos casos existe uma quantidade significativa de código que não é utilizado. Ao eliminar este código garante-se que a aplicação não vai ter módulos supérfluos na sua compilação. Assim, reduz-se significativamente o tamanho da mesma.

De forma semelhante, também podem ser removidos “selectors” de CSS que não sejam utilizados, o que é bastante comum em grandes aplicações.

2.3.1.6 Lazy Loading

Com este critério [26] é possível identificar imagens ou listas e ordená-las por importância. Assim, o browser consegue reconhecer quais são as mais importantes e carregá-las primeiro. É mais eficiente em páginas que contêm muitas imagens. A sua correta aplicação permite poupar tempo e consumo de memória, uma vez que a transferência de conteúdo é otimizada.

2.4 Estado da arte em tecnologia relevante

2.4.1 JMETER

A aplicação Apache JMeter [27] é um *software* livre, desenvolvido por Stefano Mazzocchi da fundação Apache Software, desenhada para permitir realizar testes funcionais e avaliar a performance. Esta aplicação pode ser usada para analisar e calcular a performance de aplicações Web ou de uma variedade de serviços. Testar a performance de uma aplicação Web envolve testá-la contra uma carga elevada, tráfego de utilizadores múltiplo e concorrente.

Não é viável reunir cem utilizadores com um computador e acesso à Internet para aceder simultaneamente a um site. Desta forma, esta aplicação permite simular o comportamento real de utilizadores e assim avaliar a performance do site em questão.

Podem ser enumeradas algumas vantagens oferecidas pelo JMeter:

- Licença Livre – o JMeter é completamente grátis, permitindo aos desenvolvedores usufruir do seu código fonte para o desenvolvimento.
- GUI Amigável – é extremamente fácil de utilizar e não necessita de muito tempo de adaptação.
- Independente da plataforma – é uma aplicação Java pura, operando em várias plataformas.
- Multithreading framework – exemplos concorrentes e simultâneos de diferentes funções separadas por grupos.
- Resultados – os resultados dos testes podem ser apresentados em diferentes formatos como gráficos, tabelas, árvores ou ficheiros de registo.
- Não necessita de instalação, basta executar o ficheiro *.bat.
- Extensibilidade – o utilizador pode escrever os seus próprios testes, embora a aplicação também suporte vários plugins para o mesmo efeito e para outros fins.
- Vários tipos de teste – testes de carga, teste de distribuição, testes funcionais, entre outros.
- Simular vários utilizadores com threads concorrentes submetendo a aplicação em causa a uma carga pesada.
- Suporta o uso de protocolos variados como HTTP, JDBC, LDAP, SOAP, JMS e FTP.
- Permite gravar e simular a atividade do utilizador no browser.
- Pode ser integrado com BeanShell e Selenium para a automatização de testes.

2.4.2 LoadNinja

O LoadNinja [28] é uma plataforma de testes de carga e de avaliação de performance para aplicações Web baseada na *cloud*. Ajuda os desenvolvedores e as equipas de controlo de qualidade a verificar a robustez e escalabilidade dos servidores e o seu comportamento quando enfrentam uma carga massiva. Esta aplicação permite responder às seguintes questões:

- Quantos utilizadores conseguem aceder à aplicação Web em questão sem se verificar uma alteração de velocidade percetível?
- Qual a carga que pode bloquear o servidor?
- Como varia o tempo de resposta consoante a carga aplicada?
- Como a mudança de hardware e *software* pode afetar a performance?

Com esta aplicação é possível criar e executar testes de performance diretamente no browser. Os scripts representam os cenários de utilização da aplicação, como criar uma encomenda ou procurar um artigo. O utilizador grava os scripts no navegador interno da aplicação. Basta introduzir um URL e começar a gravação. No entanto, é uma boa prática, adicionar validações para verificar se as páginas contêm o conteúdo esperado.

Um cenário é a definição do teste. Este combina um ou mais scripts, especifica o número de utilizadores virtuais, parâmetros de aumento de carga, entre outros detalhes. Para executar estes cenários, o LoadNinja executa os testes numa *cloud* com máquinas capazes de simular a carga pretendida durante o período de tempo especificado. Cada utilizador virtual usa um navegador real para repetir as ações do teste para que estes interajam com a aplicação Web da mesma forma que os utilizadores reais o fariam. Incluindo, em certos casos, alguns segundos de espera em que o utilizador estaria a ler a informação apresentada. Quando o teste estiver completo, é gerado um relatório que contém as métricas de performance recolhidas durante a execução do teste. Esta informação ajuda a fazer uma estimativa à performance da aplicação e verificar se está de acordo com os valores pretendidos.

Com o LoadNinja existe a possibilidade de gravar ações de utilizadores na aplicação a testar e depois simular as transações gravadas utilizando centenas ou milhares de utilizadores virtuais. Assim, cada utilizador virtual atua de forma muito similar a um utilizador real, isto é, os testes estão muito próximos da realidade, e não estão "apenas" a testar um URL de um site. No entanto, existem formas de aproximar os resultados um pouco mais da realidade, fazendo com que o tráfego gravado seja reproduzido em navegadores reais. Cada utilizador virtual recorre ao seu próprio navegador, não havendo interferência entre eles. Todas as ações são simuladas ao pormenor, incluindo as pausas. Se algumas páginas executam scripts no lado do cliente, o código é também executado nos navegadores dos testes.

Os testes gravados pelo LoadNinja apresentam um formato de comandos facilmente compreensível. Desta forma, a atualização ou configuração dos mesmos é rápida e sem problemas. Para simular uma quantidade considerável de utilizadores, é necessário consumir muitos recursos, e apenas um computador ou servidor, muitas vezes, não é suficiente. Esta aplicação funciona em *cloud*, pelo que o utilizador só necessita de se preocupar em escolher o número de utilizadores virtuais e a aplicação executará os testes nos seus próprios servidores, hospedados na Amazon Web Services, onde oferece também espaço para guardar os resultados. Esta aplicação permite ainda simular os utilizadores em resoluções de ecrã diferentes, como em portáteis, tablets e telemóveis.

2.4.3 NeoLoad

A NeoLoad [29] é uma plataforma automatizada de testes de performance para empresas que testam continuamente as suas aplicações. Fornece aos desenvolvedores a manutenção de testes automáticos e uma forma realista de simulação do comportamento de utilizadores. Foi desenhada, desenvolvida e comercializada pela Neotys, uma empresa situada em Gémenos, França.

Esta aplicação simula tráfego de utilizadores (pode chegar aos milhões) para determinar a performance da aplicação sob carga, analisa tempos de resposta e identifica o número de utilizadores simultâneos que esta consegue suportar. Os testes podem ser efetuados dentro da *firewall* (internamente) ou através da *cloud*.

Além de simular tráfego de rede, também simula as atividades de transação por parte do utilizador, incluindo tarefas comuns como submeter formulários ou executar procuras simulando utilizadores "virtuais" a aceder a certos módulos de aplicações Web. Fornece ainda informação detalhada sobre a performance das aplicações que indica como resolver problemas de congestionamento e excesso de carga. Monitoriza as mais recentes aplicações Web, base de dados e servidores como Jboss, HP-UX 11, Weblogic, WebSphere, Apache Tomcat e a base de dados MySQL.

Os scripts são desenvolvidos através da sua interface gráfica que fornece condições, *loops* e outras estruturas de programação. Pode ser inserido *JavaScript* para uso mais avançado, tal como encriptação de palavras passe. De seguida, enumeram-se algumas das vantagens desta plataforma.

- Suporta autenticação básica, NTLM e baseada em formulários.
- Suporta qualquer tipo de aplicação Web incluindo os que usem J2EE, .NET, AJAX, Flex, Silverlight, GWT, SOAP, PHP, entre outros, desde que atuem em conformidade com HTTP 1.0 ou 1.1.
- Suporta aplicações móveis híbridas e nativas.
- JSON e SPDY também são suportados.
- Consegue simular condições da rede (e.g., latência).

2.4.4 Blazemeter

O Blazemeter [30] fornece os seus serviços através de testes de carga e de performance. Oferece uma interface para a criação de testes de carga estáticos e consegue executar qualquer script de JMeter para testes de carga dinâmicos. Um dos seus pontos fortes é a implementação de JMeter com caraterísticas apropriadas para empresas como a distribuição de testes, relatórios em tempo real, integração com ferramentas de desenvolvimento para integração contínua e monitorização da performance de aplicações. Pode identificar-se as seguintes vantagens desta plataforma.

- Compatibilidade com JMeter.
- Criação de testes até um milhão de utilizadores concorrentes.

- Configuração de testes em minutos.
- Execução de testes sem necessidade de scripts ou, simplesmente, introduzir o URL do *website*.
- Execução interna ou na *cloud*.
- Execução de teste em diferentes locais geográficos.
- Simulação de testes mobile a partir de dispositivos reais.
- Facilita a colaboração entre a equipa de trabalho com a partilha de scripts e relatórios.
- Integração com as principais ferramentas de CI e aplicações de monitorização de performance.
- Suporte e serviços profissionais.
- Resultados de testes em relatórios detalhados e em tempo real.

2.4.5 Flood

A Tricentis é uma empresa que desenvolveu a aplicação Flood [31]. Esta aplicação é uma plataforma de testes de performance que permite executá-los globalmente distribuídos usando ferramentas *open source*, que incluem JMeter, Gatling e Selenium. Os testes desta aplicação podem ser escalados para concorrência e com uma taxa de transação máxima em qualquer momento. Fornece tratamento da infraestrutura e cria relatórios agregados e em tempo real. Esta aplicação apresenta as seguintes vantagens.

- Simula milhares de utilizadores de forma autónoma e expande-se sob pedido até centenas de milhares de utilizadores.
- Executa testes com um URL simples ou planos de testes mais avançados que são compatíveis com JMeter ou Gatling.
- Relatórios precisos com descrição de estatísticas a um nível individual de transações e com dados disponíveis em formatos JSON ou CSV.

2.4.6 Gatling Frontline

A Gatling Frontline [32] é uma framework livre de testes de carga e performance baseada em Scala, Akka e Netty. O *software* foi desenhado para ser usado como uma ferramenta de testes para analisar e medir a performance de uma variedade de serviços com o foco em aplicações Web. Desenhada para facilitar testes contínuos, integra-se com ferramentas de compilação e oferece um gravador de páginas Web, bem como relatórios elaborados.

- Gravador independente de proxy HTTP.
- Linguagem de domínio específica (DSL) para desenvolvimento de testes fácil de compreender.
- Mecanismo assíncrono e sem bloqueios para performance máxima.
- Suporte excelente de protocolos HTTP(s).
- Relatório HTML de fácil compreensão com os resultados dos testes efetuados.

2.4.7 Load Impact

O Load Impact [33] é um serviço em nuvem para determinar o desempenho de *websites* e APIs, fornecendo as ferramentas necessárias ao negócio para que o desempenho da aplicação seja otimizado. Os seus casos de uso incluem testes locais, testes de regressão, testes na nuvem, entre outros. O seu uso pode ser através de testes elaborados em *JavaScript* ou usando a interface gráfica do Load Impact que gera o código necessário para o teste. Pode-se ainda converter coleções do Postman [34] e testes do JMeter. Usa-se a linha de comandos para executar os testes localmente ou na nuvem e analisa-se os seus resultados. Podem ser enumeradas algumas vantagens oferecidas por este serviço.

- Testes em grande escala, até 1.2 milhões de utilizadores.
- Sem licenças ou *software* para instalar.
- Suporta qualquer aplicação ou serviço baseado em protocolos HTTP.
- Monitorização de servidor.
- Gravador de cenário simples.
- Simulador de browser.
- Dados em tempo real.

2.4.8 LoadView

Com o LoadView [35], desenvolvido por Dotcom-Monitor, é possível demonstrar o desempenho atual das aplicações sob carga, tal como os utilizadores as experienciam. Este *software* utiliza testes de carga baseados em browsers reais para *websites*, aplicações Web e APIs. É com facilidade que são criados scripts que simulam a interação de utilizadores com o *website* ou uma aplicação usando um gravador Web, ou até editando manualmente os scripts com código C#. De seguida, apresentam-se algumas das vantagens desta aplicação.

- Testes de performance em browsers reais através de *cloud*.
- Suporta RIA (Rich Internet Applications), como o Flash, Silverlight, Java, HTML5, PHP, Ruby, entre outros.
- A criação de scripts de teste é fácil e rápida sem a necessidade de escrever código.
- Compatibilidade com mais de quarenta browsers e dispositivos móveis.
- Mais de treze localizações de *clouds* distribuídas pelo mundo inteiro usando os serviços da Amazon e Google Cloud Platform.
- Identifica os limites da aplicação e assegura a sua escalabilidade.
- Relatórios e métricas de performance que podem ser partilhados com vários colaboradores internos.

2.4.9 OctoPerf

O OctoPerf [36] é uma plataforma de testes de desempenho *full-stack* na nuvem ou localmente. Permite simular milhares de utilizadores concorrentes em qualquer site ou aplicação móvel. Este *software* apresenta uma interface Web que permite definir a política de carga, executar os

testes e analisá-los diretamente de um browser. O OctoPerf foi desenvolvido por uma equipa que se foca, principalmente no suporte ao utilizador com documentação contextual, tutoriais em vídeo, uma metodologia de treino e um *live chat.* Os seguintes pontos representam algumas vantagens que esta plataforma oferece.

- Suporte total a JMeter.
- Mecanismo de scripts sem necessidade de escrever código.
- Injeção de carga na nuvem e localmente.
- Mecanismo de monitorização de servidor ilimitado e grátis.
- Relatórios de análise ao vivo, customizáveis e exportáveis.
- Tendências e comparação de resultados.
- Gravar e desenhar comportamentos de utilizadores virtuais.
- Múltiplas localizações geográficas.

2.4.10 Taurus

Criado pela Blazemeter, o Taurus [37] é uma estrutura grátis que oferece uma solução simples para criar e executar testes de performance, bem como a integração com *software* de teste como Selenium, Gatling ou JMeter. Esta aplicação utiliza ficheiros YAML para a criação dos testes, o que os torna de fácil compreensão. Podem-se enumerar as algumas vantagens.

- Executar testes de JMeter já existentes.
- Juntar vários scripts de testes num único cenário.
- Relatório em tempo real.

2.4.11 Smartmeter.io

O SmarteMeter.io [38] é uma ferramenta de testes multiplataforma caracterizada pela rapidez e facilidade de criação e execução de testes, gestão dos mesmos e geração de relatórios dos resultados. É baseado no JMeter mas adiciona novas funcionalidades como relatórios aprofundados, gravador de cenários avançado, modo distribuído intuitivo e simples, critérios de aceitação, entre muitas outras.

Relatórios de teste gerados automaticamente com detalhes e resultados contendo:

- Visão geral de gráficos e estatísticas.
- Critérios de aceitação.
- Análise de tendências.
- Ferramenta de comparação de gráficos.
- Configurações de teste.
- Ficheiros CSV para análise adicional.
- Sem configurações de *proxy* ou *plugins.*
- Transações e tempos de espera automáticos.
- Transações e tempos de espera automáticos.

- Simulação de browser mobile.
- Separação de recursos estáticos e dinâmicos em pedidos.
- Registos detalhados de pedidos e respostas HTTP.

Pode-se reproduzir inúmeros utilizadores virtuais de diferentes localizações instalando geradores de carga em múltiplas máquinas. As funcionalidades adicionais incluem:

- Uma vez instalado estará completamente automatizado.
- Servidor de dados incluído.
- Resultados em tempo real.
- Monitorização num ambiente automático (CPU, RAM, HDD).
- Combinação de nuvem e geradores de carga locais.

Quando houver alterações significativas nos tempos de resposta, significa que poderá haver um problema no sistema testado. O SmartMeter monitoriza recursos para assegurar que não haja insuficiência dos mesmos no ambiente de teste e não sejam a causa dos tempos de resposta mais longos.

2.4.12 WebLOAD

O WebLOAD [39] é uma ferramenta de testes de carga e performance a nível empresarial. É a ferramenta de escolha de várias empresas com uma grande carga de utilizadores e requisitos de teste complexos. Permite executar testes de stress em qualquer aplicação Web gerando a carga a partir da *cloud* e de máquina locais. Os seus pontos fortes são a sua flexibilidade e facilidade de uso, possibilitando a definição dos testes necessários com funcionalidades como correlação automática e scripts em JavaScript. A ferramenta proporciona uma análise clara do desempenho da aplicação, identificando problemas e limites que podem impedir que se cumpram objetivos. Esta aplicação suporta centenas de tecnologias, desde protocolos Web a aplicações de empresas e tem integração com Jenkins, Selenium e muitas outras ferramentas para habilitar testes contínuos para DevOps. Podem ser enumeradas as seguintes vantagens.

- Correlação, parametrização, validação de respostas, mensagens, JavaScript nativo e debbuging.
- A consola de geração de carga gera inúmeros utilizadores virtuais localmente e na *cloud*.
- Os painéis de análise oferecem mais de oitenta templates de relatórios configuráveis e permitem a partilha entre várias equipas.
- Integrações permitem o uso do WebLOAD junto com ferramentas APM, *software* livre (Selenium,Jenkins), testes móveis, etc.

2.4.13 GTmetrix

O relatório dos resultados apresentado pelo GTmetrix [21] resume claramente a performance das páginas baseadas em indicadores chave do seu tempo de carregamento. O acompanhamento do desempenho da página é simples com monitorização agendada e

possibilidade de ver os resultados também através de gráficos interativos. Podem-se configurar alertas para notificar quando a página apresentar resultados abaixo do esperado.

É possível reproduzir o carregamento das páginas em vídeo e identificar exatamente onde acontecem os pontos de maior constrangimento. Podem enumerar-se as funcionalidades principais desta ferramenta.

- Analisa a página com as regras do GooglePageSpeed e Yahoo YSlow.
- Obtém o tempo de carregamento da página, o tamanho total da página e o número total de pedidos.
- Compara a performance da página com a média de todos os sites analisados com o GTmetrix.
- Monitorizar páginas e executar testes diários, semanais ou mensais para assegurar um desempenho ótimo.
- Visualizar performance com gráficos diferentes: tempo de carregamento da página, tamanho da página e respetivos pedidos ou resultados de PageSpeed e YSlow.
- Definir um intervalo de tempo para avaliar o histórico do desempenho.
- Anotar áreas de interesse nos gráficos e introduzir contexto nos dados.
- Definir alertas com várias condições baseadas nos resultados de PageSpeed/YSlow, tempo de carregamento, tamanho, etc.
- Diminuir a velocidade de reprodução do vídeo até quatro vezes a velocidade original e determinar os problemas de carregamento.
- Avançar para os principais objetivos de carregamento da página como o primeiro Byte e o carregamento do DOM.
- Guardar vídeos de carregamento da página onde são acionados alertas definidos.
- Ver o carregamento da página frame por frame.

2.4.14 PageSpeed Insights (PSI)

Desenvolvido pela Google, o PSI [22] avalia a performance de páginas da Web em dispositivos móveis e computadores e apresenta sugestões sobre como melhorar o seu desempenho. Os dados que resultam da sua análise são divididos em dois tipos de informação. O primeiro tipo, informação de laboratório, é informação recolhida num ambiente controlado com um conjunto de dispositivos e definições de rede já configurados. Desta forma, é possível encontrar problemas diretamente relacionados com a performance da página, no entanto pode não representar problemas que afetam o utilizador. Para isso, existe o segundo tipo, informação de campo, isto é, monitorização de utilizadores reais guardando informações de carregamento da página. Assim, consegue-se capturar resultados mais próximos da experiência dos utilizadores.

2.4.15 Lighthouse

O Lighthouse [40] é uma ferramenta automatizada e *open-source,* também desenvolvida pela Google com o objetivo de melhorar as páginas Web. É possível executá-la em qualquer página, seja pública ou com autenticação. Permite revisões de desempenho, acessibilidade, entre outras. Este *software* pode ser executado através do Chrome DevTools, na linha de comandos e existe a possibilidade de ser instalado como um módulo de Node. O Lighthouse verifica um URL e faz a sua avaliação, gerando um relatório sobre o desempenho da página que contém informação detalhada sobre como melhorar cada aspeto analisado.

2.5 Análise de tecnologias

As tecnologias anteriormente descritas foram testadas e passaram por um período de experimentação, onde foi possível retirar ilações que permitiram avaliar e entender quais se adequavam melhor às necessidades do projeto. A tabela 1 apresenta uma comparação de algumas características das ferramentas analisadas.

Tabela 1 - Comparação de algumas características das ferramentas analisadas

	Gratuito	GUI	Relatórios detalhados e compreensíveis	Cloud	Local	Utilizadores concorrentes "ilimitados"	Diferentes localizações geográficas
Jmeter	x	x	x		x		
LoadNinja	—	—	—	—	—	—	—
NeoLoad		x	x	x	x	x	x
Blazemeter		x	x	x	x	x	x
Flood		x	x	x	x	x	x
Gatling	x		x		x		
LoadImpact		x	x	x		x	x
LoadView		x	x	x			x
OctoPerf			x	x	x	x	
Taurus	x		x		x		
SmartMeter		x	x	x	x	x	x
WebLOAD		x	x	x	x	x	
GTmetrix*	x	x	x	x			x
PSI*	x	x	x	x			
Lighthouse*	x	x	x	x	x		

* Ferramentas que apenas permitem avaliar páginas singulares

Como se pode verificar na tabela 1, as aplicações gratuitas ficam em clara desvantagem em relação às que oferecem um serviço pago. No entanto, as potencialidades oferecidas, justificam convenientemente o seu uso. Das que permitem o uso de serviços limitados ou períodos gratuitos, as que mais justificam a sua utilização são o Flood e o Blazemeter. No entanto, como

nenhuma permite efetuar testes locais como parte dos seus serviços limitados, este projeto terá como principal fonte de avaliação as tecnologias gratuitas oferecidas por Jmeter, Taurus e Lighthouse.

3 Avaliação de soluções/abordagens existentes

As soluções e abordagens apresentadas no capítulo Contexto e Estado da arte podem ser divididas em metodologias e tecnologias. As tecnologias correspondem a aplicações ou *software* responsável por ajudar a identificar problemas no desempenho da aplicação em causa. Estas podem submeter a aplicação a vários testes contínuos que permitem identificar onde ocorrem pontos de maior sobrecarga e que podem atrasar as respostas da mesma, ou então podem só avaliar uma página individual identificando aspetos cruciais que possam prejudicar o desempenho da aplicação naquele ponto em particular. As metodologias são técnicas e procedimentos que podem ajudar a corrigir os problemas identificados pelas tecnologias. O problema de performance na Web é comum e várias adversidades possuem uma solução conhecida. Assim, quando as tecnologias reconhecem certos erros ou pontos em que poderá haver uma melhoria, podem ser usadas metodologias relacionadas.

Todas as tecnologias aqui descritas foram testadas e examinadas com o intuito de identificar quais as que melhor se enquadravam ao problema descrito. Algumas das aplicações são de uso livre enquanto outras são pagas. As pagas, apresentavam uma versão grátis, sendo esta limitada a nível de funcionalidades ou em termos de tempo.

As avaliações de desempenho de uma aplicação dependem de muitos fatores como a rede, a máquina do cliente e servidor, e estes resultados podem variar. Optou-se pela utilização de múltiplas ferramentas de forma a ser possível identificar onde o problema mais incide. Desta forma, no final da avaliação, haverá maior informação sobre os resultados.

Para obter uma decisão relativamente ao *software* a ser utilizado, foi considerada a procura de serviços que não apresentassem um custo adicional à empresa. No entanto, foram testadas as versões gratuitas de cada aplicação com o intuito de complementar e expandir o conhecimento acerca desses produtos. Na eventualidade de ser encontrado um que justificasse a sua compra, seria comunicado aos responsáveis da empresa qualificados para a tomada de decisão.

Acerca do *software* gratuito existente no mercado, o grande obstáculo seria a gestão de recursos do computador. Uma vez que grande parte dos testes efetuados por este tipo de *software* envolve a simulação de utilizadores virtuais e aumento de carga nas aplicações, é muito importante que os resultados não sejam degradados pelo mau desempenho da máquina. Certas aplicações demonstraram melhor rendimento e uma melhor gestão desses recursos, bem como resultados mais compreensíveis e detalhados, o que foram considerados fatores cruciais na decisão.

Para poder apurar quais os programas que apresentavam mais benefícios e que justificavam o seu uso, durante a elaboração do projeto foram testadas certas aplicações, escolhidas com

critério para se obterem diferentes resultados. Foram efetuados testes de carga e de stress, bem como foi realizada a avaliação, usando certas métricas adequadas às seguintes aplicações:

- Atendimento – aplicação da Glintt onde este trabalho incide devido ao seu desempenho limitado.
- *Aliexpress* – aplicação de compras *online* que aplica algumas métricas de otimização de desempenho definidas anteriormente e apresenta resultados bastante positivos [41].
- *Amazon* – aplicação homóloga à anterior.
- MaisFutebol – site de notícias desportivas que apresenta valores de desempenho muito abaixo do esperado.

Tanto o Atendimento como o site MaisFutebol carecem de um aperfeiçoamento a nível de desempenho, contrariamente ao *Aliexpress* e *Amazon* que apresentam valores muito perto da excelência. Estes valores positivos devem-se à utilização de métricas de otimização e grande parte destas estão identificadas neste documento.

As tecnologias mais apropriadas e que mais justificaram o seu uso foram o Jmeter e Taurus, nas aplicações grátis, Flood e Blazemeter nas aplicações pagas, mas que apresentam serviços grátis limitados, e por fim, as aplicações GTmetrix e Lighthouse para avaliar o desempenho de páginas individualmente.

Relativamente às aplicações grátis, considerou-se o Jmeter por ser um *software* que permite efetuar vários tipos de testes de fácil configuração, apresenta uma interface intuitiva e os resultados podem ser apresentados em diversos formatos para facilitar a compreensão do utilizador. Uma vez que várias aplicações concorrentes foram baseadas no Jmeter, os seus ficheiros de teste, bem como os resultados gerados automaticamente, são compatíveis com essas novas aplicações. Como se pode observar na figura 5, o Jmeter avalia cada pedido efetuado pelo site individualmente e, assim que cada utilizador virtual termina a sua pesquisa, a informação é guardada num rótulo designado por “Test”. É possível observar ainda vários valores relativos a cada tarefa como o tempo médio de carregamento, o valor mínimo, máximo, entre outros.

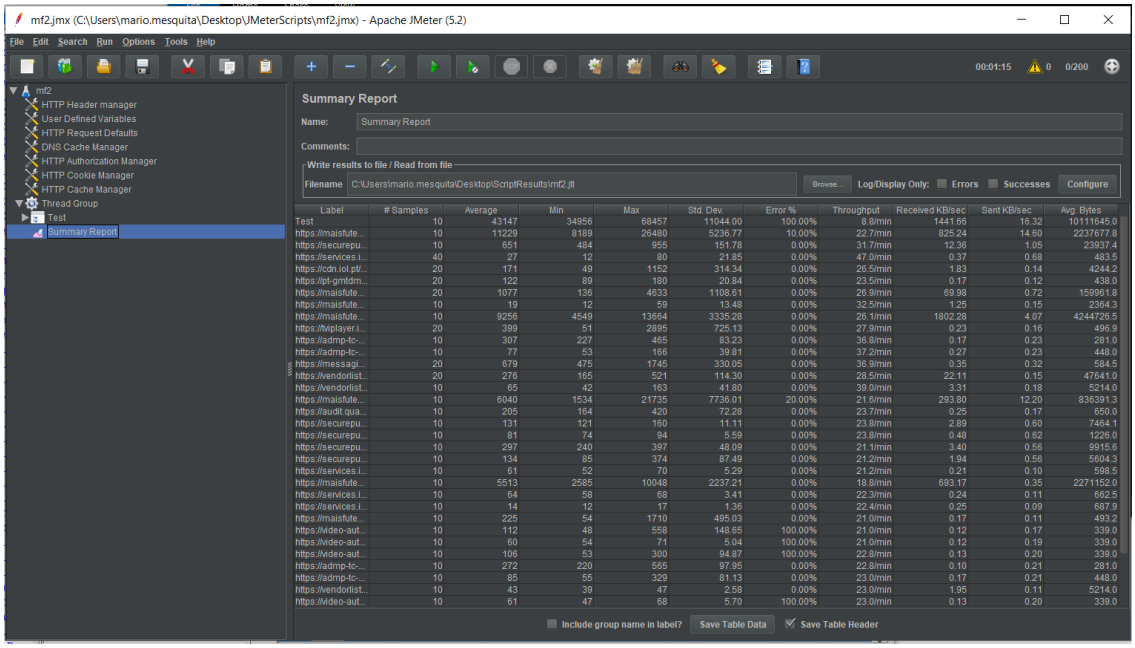


Figura 5 - Exemplo dos resultados de um teste JMeter

A segunda aplicação grátis que também justificou a sua utilização foi o Taurus. Com esta aplicação era possível reutilizar os testes do Jmeter, uma vez que são compatíveis, e executá-los na linha de comandos do computador, evitando assim a perda de recursos com interfaces gráficas. Um fator importante foi a possibilidade de analisar os resultados através da interface do Blazemeter, uma vez que foram desenvolvidos pela mesma empresa.

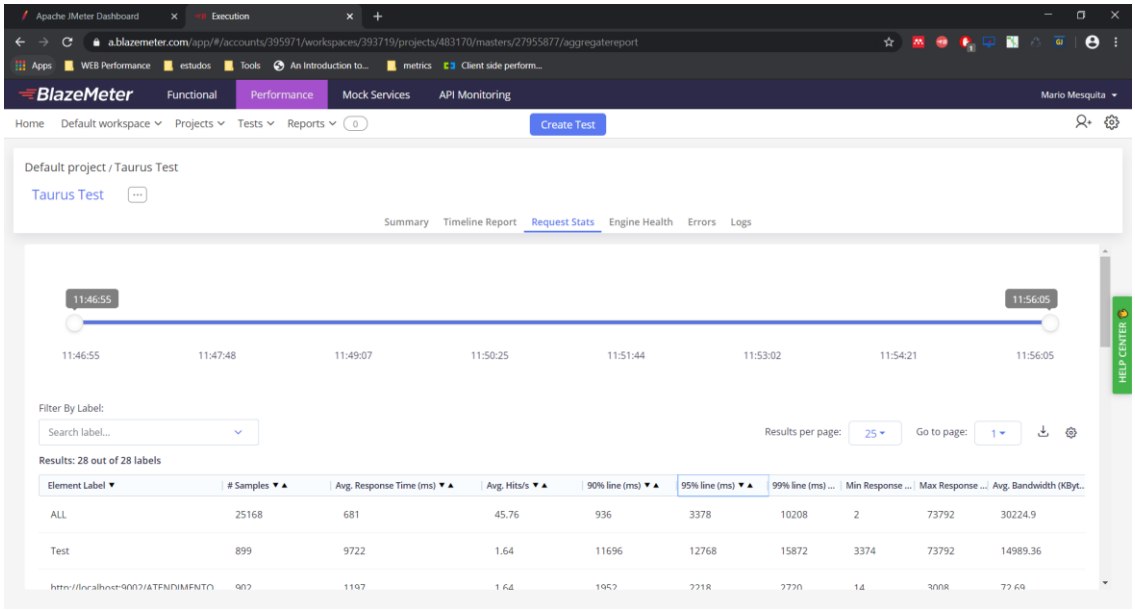


Figura 6 - Exemplo de relatório de um teste Taurus com interface Blazemeter

Na figura 6 está representado o relatório de um teste efetuado pelo Taurus cujos resultados são caracterizados de forma semelhante ao Jmeter, onde os pedidos são analisados

individualmente e depois agrupados num rótulo designado “Test” que vai conter a informação da tarefa que o utilizador virtual completou.

Em relação a tecnologias pagas, o Blazemeter possui uma documentação excelente que permitiu a fácil compreensão do seu funcionamento. Como permite a utilização do seu *software* com um número limitado de testes por mês e 50 utilizadores concorrentes por teste, foi possível avaliar alguns sites e concluir que os relatórios apresentados mostravam bastante detalhe e informação útil. Inclui-se a figura 6 na análise deste programa uma vez que a interface é semelhante.

Por outro lado, o Flood foi a única aplicação paga que permitiu efetuar testes com mais de 100 utilizadores virtuais concorrentes no período experimental. Na figura 7 pode-se observar o resultado de um teste com 500 utilizadores. Embora a sua interface não se apresente com tanto detalhe como o Blazemeter, ainda possui uma grande quantidade de informação.

Figura 7 - Reslutado de um teste Flood

O GTmetrix é uma ferramenta de teste de velocidade de um website e permite compará-lo com milhares de outros sites. Esta ferramenta possui um algoritmo de pontuação que não só demonstra os problemas encontrados na sua análise como também sugere formas de os corrigir, como se pode verificar na figura 8.

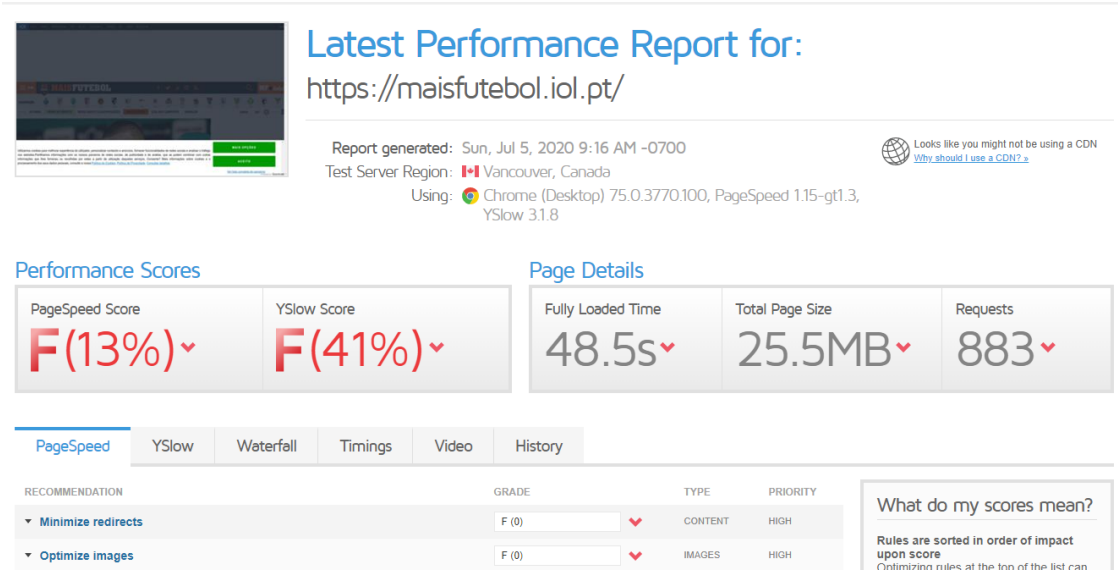


Figura 8 - Resultado de um teste GTmetrix

O Lighthouse é uma ferramenta de análise de desempenho de páginas web presente no navegador Google Chrome. Esta ferramenta permite avaliar o estado de uma página através do seu próprio sistema de pontuação, analisando diferentes métricas e calculando o seu valor final. Como se pode observar na figura 9, é gerado um relatório onde são identificados os problemas associados à página analisada, bem como sugestões para melhorar os resultados de desempenho.

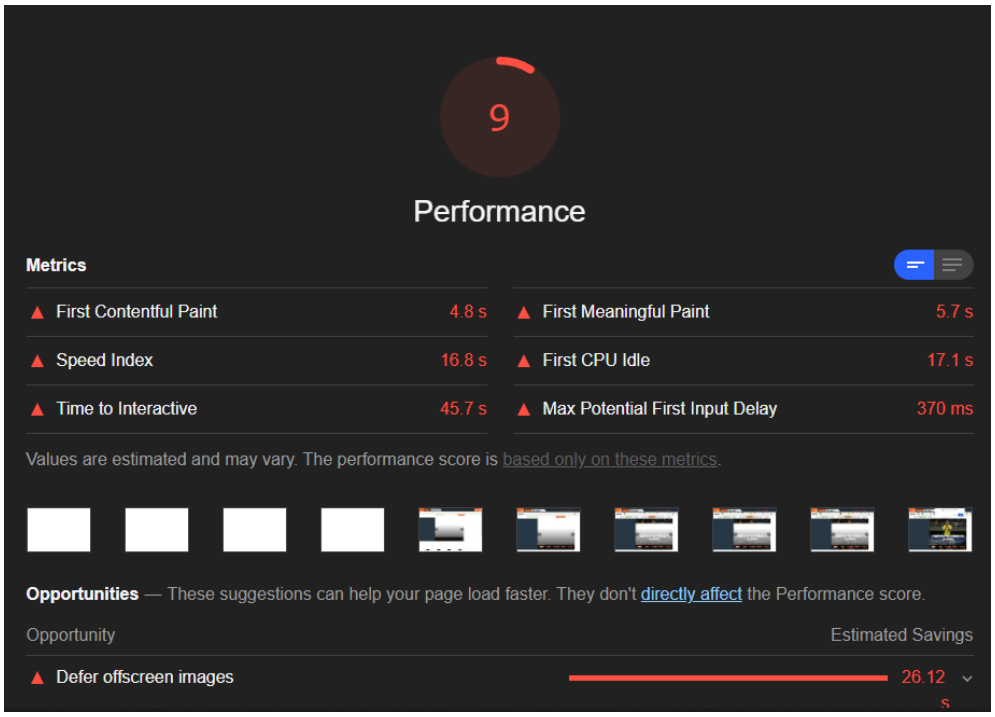


Figura 9 - Exemplo de um relatório Lighthouse

Podem verificar-se imagens relativas a todas a outras aplicações estudadas no Anexo A no final deste documento.

As metodologias identificadas são soluções apontadas para os problemas mais frequentes, pelo que devem ser postas em prática na maior parte dos casos encontrados neste projeto. Estas serão mais aprofundadas e avaliadas no Capítulo 5 - Avaliação, quando se verificarem os resultados da sua utilização.

4 Design

A solução passa pela definição de uma metodologia de análise de desempenho que permita o uso de ferramentas de avaliação de performance das aplicações da empresa, como é o caso da criação, execução e avaliação de testes de carga automatizados. Os resultados destes testes, conjuntamente com os de outros tipos de teste, apresentam sempre valores diferentes, sendo necessário analisá-los com atenção e repeti-los várias vezes até ser possível obter resultados mais generalizados. Por isso, a necessidade da análise exaustiva anteriormente referida. Após esta análise, existe um conjunto de processos com determinadas metodologias, que, dependendo da aplicação em questão, são postos em prática com vista a corrigir os problemas identificados. É um processo longo e tem-se em conta que algumas metodologias podem não ter tanto impacto como outras. No entanto, só com o estudo e avaliação das mesmas é que se pode comprovar a sua eficiência. O produto resultante é novamente testado e o processo repete-se até se verificarem resultados positivos. Entendeu-se que a melhor forma de avaliar o novo desempenho, para além da sua reavaliação, passa pelo uso de inquéritos aos desenvolvedores da aplicação, uma vez que têm total conhecimento do seu funcionamento e rapidamente identificam mudanças benéficas na sua performance. A figura 10 caracteriza o processo de otimização que se estabeleceu no desenvolvimento deste projeto.

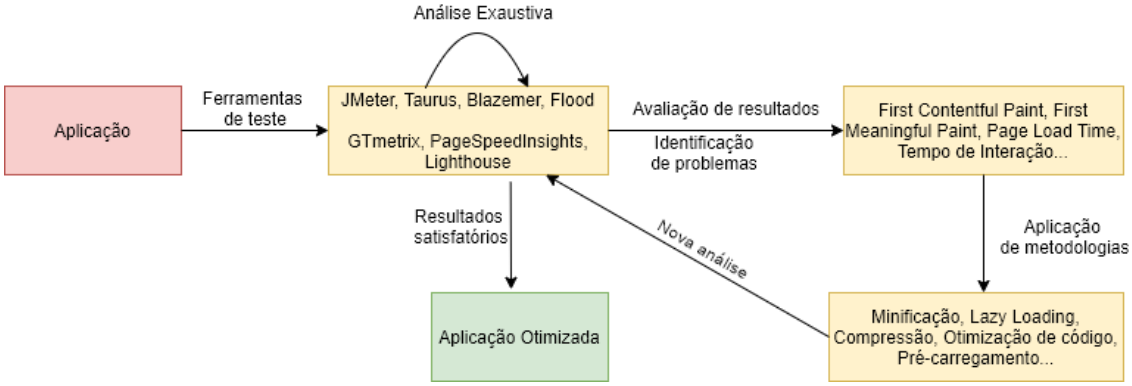


Figura 10 - Processo de Otimização

A aplicação à qual se pretendeu submeter este estudo denomina-se de Atendimento (ATD), que conta com vários projetos. Esta aplicação Web apresenta vários serviços no âmbito da gestão hospitalar. Como tal, existem módulos de Gestão de Paciente, Gestão de Agendamento para marcação de consultas, Internamento, Consulta Externa, bem como a Faturação, entre outros. Todos estes módulos tornam a aplicação pesada e à medida que são adicionados mais módulos, torna-se cada vez mais lenta. Pelo exposto, foi considerada uma aplicação com uma necessidade evidente de melhoria de desempenho.

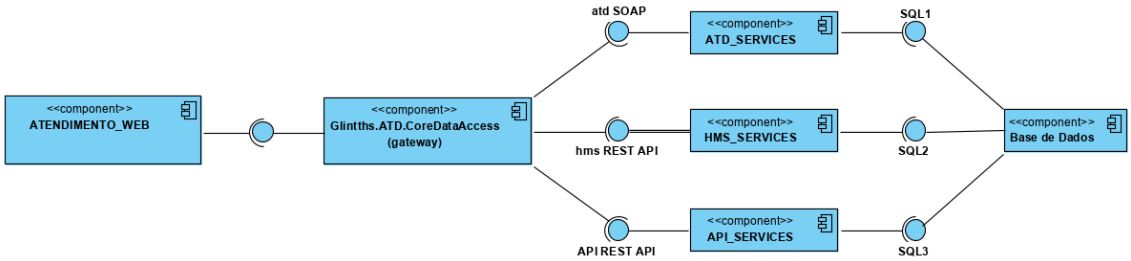


Figura 11 - Diagrama de Componentes Do Atendimento

A figura 11 representa o Diagrama de Componentes do Atendimento. A aplicação envia informação ao *gateway "*Glinths.ATD.CoreDataAccess" de modo a este comunicar com os serviços requisitados do "ATD_SERVICES", "HMS_SERVICES" e "API_SERVICES". Por sua vez, estes serviços são responsáveis por tratar a informação da base de dados.

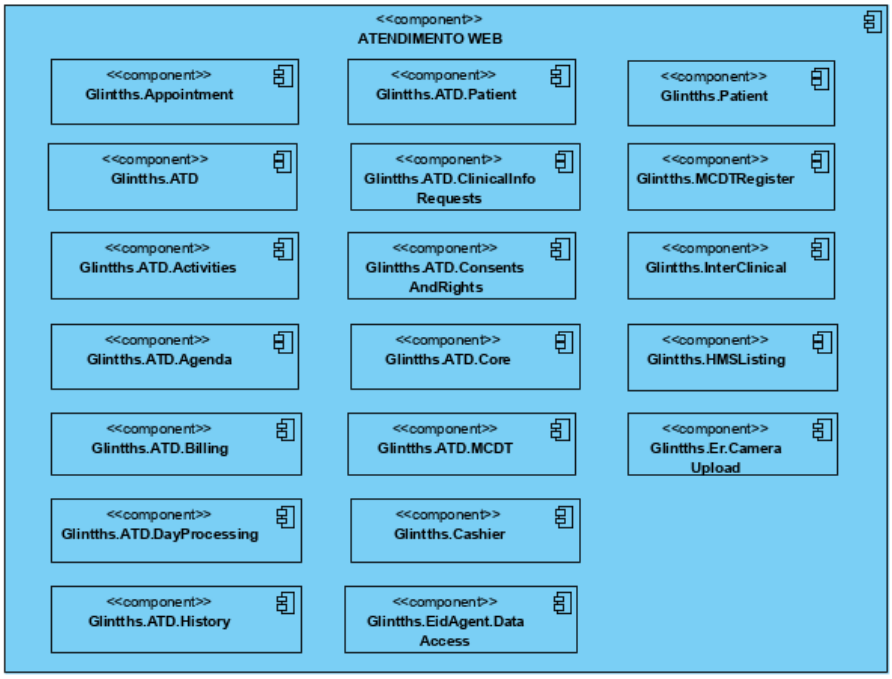


Figura 12 - Componente Atendimento

A figura 12 tem como objetivo evidenciar os módulos da aplicação. Como referido anteriormente, existem muitos módulos dentro da aplicação e cada um apresenta o seu nível de complexidade. Assim sendo, torna-se difícil para o sistema cumprir alguns requisitos não funcionais como a disponibilidade, eficiência e performance.

Esta aplicação encontra-se alojada nos servidores de Cloud da Oracle. A empresa fornece plataformas, infraestruturas, *software* e dados, como serviços pagos. Através destes serviços é possível criar, implementar e integrar aplicações. São suportados inúmeros padrões e linguagens de programação, bem como tipos de base de dados, ferramentas e estruturas.

A base de dados desta aplicação é relacional (SQL), ou seja, define relacionamentos sob a forma de tabelas. A programação SQL facilita o uso de funções CRUD (Create, Read, Update, Delete), no entanto, vai para além disso, com o auxílio na otimização e manutenção da base de dados [42]. Pode ainda dizer-se que é verticalmente escalável, ou seja, de modo a aguentar com tráfego elevado, é possível aumentar os atributos de um único servidor (CPU, RAM, SSD, etc.).

Existe ainda outro tipo de base de dados em que a aplicação poderia ser desenvolvida, que seria NoSQL, ou base de dados não relacional. Este tipo de base de dados não requer um esquema fixo, é horizontalmente escalável (de modo a suportar o aumento de tráfego podem ser adicionados mais servidores) e evita ligações.

Numa aplicação com um grande número de transações e com serviços mais pesados, as bases de dados relacionais são mais recomendadas pois seguem as propriedades ACID [43] que garantem Atomicidade, Consistência, Isolação e Durabilidade. As bases de dados não relacionais normalmente seguem as propriedades BASE (Basically Available, Soft State, Eventually Consistent) [44]. Como se pode verificar na figura 13.

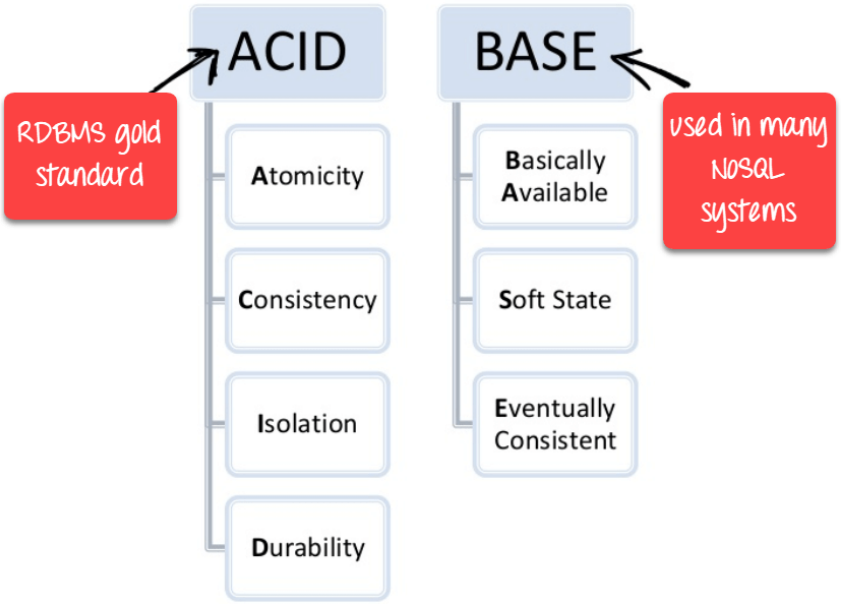


Figura 13 - Propriedades ACID vs Propriedades BASE (retirado de [42])

Depois de avaliadas as alternativas, optou-se por se recorrer a uma base de dados relacional não só por se adaptar melhor às necessidades do projeto, como também por já se utilizarem os serviços semelhantes noutros produtos da empresa.

5 Experimentação e Avaliação

Para realizar uma avaliação da solução e de modo a obter uma compreensão total do problema, foram identificadas algumas métricas que indicam valores importantes na avaliação do desempenho das aplicações. Atendendo aos resultados dessas métricas, é possível verificar o comportamento atual das páginas de um site e identificar onde se foca grande parte dos problemas a ele associados. Com esta informação, é possível aplicar as metodologias anteriormente descritas, de forma a assegurar um melhor comportamento, bem como a melhoria nos resultados das métricas usadas.

5.1 Métricas

De seguida, são apresentadas as métricas usadas no desenvolvimento deste projeto.

5.1.1 First Paint (FP) and First Contentful Paint (FCP)

Estas métricas [45] correspondem ao ponto imediatamente a seguir à navegação, quando o browser apresenta no ecrã os primeiros pixéis. A principal diferença entre estas duas métricas é que, no caso do FP, representa o momento em que o ecrã exibe algo visivelmente diferente ao que estava antes, enquanto o FCP representa o momento em que é apresentado no ecrã o primeiro fragmento de conteúdo relevante (como uma imagem ou texto).

5.1.2 First Meaningful Paint (FMP)

FMP [46] avalia o tempo que o conteúdo principal demorou a aparecer na página. Por exemplo, no caso do Youtube, corresponderia ao vídeo principal. Numa aplicação de meteorologia, seria a previsão de tempo de uma localização. Grande parte das páginas Web apresentam partes que são mais relevantes que outras, portanto se estas forem carregadas primeiro o utilizador pode nem se aperceber que o resto da página ainda está a ser carregada.

5.1.3 Tempo de Carregamento da Página

Uma das métricas mais importantes na monitorização do desempenho na Web é o tempo de carregamento da página [47]. Uma questão de milissegundos pode significar uma grande perda ou um grande ganho financeiro. O tempo de carregamento de uma página, calcula o período desde que o utilizador clica num link ou procura um endereço, até ao momento em que a página se encontra completamente carregada.

5.1.4 Tempo para o título

O intervalo de tempo entre o momento em que um utilizador faz o pedido de um *website* e o momento em que o título do site aparece na barra de separador, é denominado de "*Time to title*" [48]. Apresentar o título o mais rapidamente possível, transmite ao utilizador que o site é rápido e legítimo, o que poderá fazer com que este seja mais tolerável com o tempo de espera.

5.1.5 Start Render Time (STR)

Esta métrica [49] representa o primeiro momento em que um elemento é apresentado no ecrã, mesmo antes do conteúdo da página começar a ser carregado. Pode ser algo tão simples como uma cor de fundo, mas é a primeira indicação de que algo está a acontecer. A diferença entre esta métrica e o FP é a forma como são avaliadas. O SRT é calculado avaliando o vídeo do carregamento *frame* a *frame* até se determinar que é apresentado algo que não seja uma página em branco, enquanto o FP é o próprio browser que indica quando considera que está a apresentar o primeiro pedaço de conteúdo. Este último consegue ser muito preciso, mas por vezes calcula o valor quando apresenta a página em branco.

5.1.6 Taxa de Rejeição

Esta métrica [50] avalia a percentagem de visitantes que abandonam o site após terem visto apenas uma página. Uma taxa alta normalmente indica que os visitantes estão a visitar a página, mas a baixas velocidades, conteúdo desinteressante e um design pouco atraente, podem estar a prejudicar a experiência do utilizador.

5.1.7 Tempo de Interação

O tempo de interação [51] é o intervalo de tempo entre o pedido do utilizador e o momento em que este pode interagir com a página, como clicar em links, escrever em caixas de texto ou realizar outras ações. No entanto, a página não necessita de estar completamente carregada pelo que pode haver elementos que ainda estejam a ser processados. Esta métrica identifica o momento em que a aplicação está simultaneamente visível e capaz de responder à entrada de dados do utilizador.

5.1.8 Pedidos por segundo

Como o próprio nome indica, esta métrica [52] assinala o número de pedidos que são enviados ao servidor a cada segundo. Um pedido pode ser considerado como qualquer interação com os recursos da página, como as páginas HTML, imagens, ficheiros, consultas à base de dados, entre outros.

5.1.9 Peso Geral

O peso geral [53] de um site é caracterizado pela quantidade de bytes que o utilizador recebe. Tendo em conta que as páginas Web têm crescido exponencialmente nos últimos anos, é cada vez mais importante avaliar este aspeto.

5.1.10 Taxa de Erro

A taxa de erro [54] é a percentagem de pedidos em que houve problemas. Quando se verifica um pico na taxa de erro num determinado momento de um teste de carga, geralmente significa que existe algo que está a impedir que a aplicação funcione corretamente.

5.1.11 Pico de Tempo de Resposta

Esta métrica [55] avalia anormalidades dentro do tempo médio de resposta, apresentando elementos que demoram mais tempo do que o normal a carregar. Com esta avaliação torna-se muito mais fácil identificar aplicações que apresentam resultados mais lentos que o suposto.

5.1.12 Tempo de Conexão

O intervalo de tempo entre um pedido e o momento em que se estabelece a conexão entre o browser do utilizador e o respetivo servidor, é designado de tempo de conexão [56]. É difícil identificar os problemas que causam um atraso no tempo de conexão pois estes podem depender de muitos fatores. Muito tráfego no servidor pode ser uma das causas que faz com que os tempos disparem.

5.1.13 Tempo Total de Bloqueio

O tempo total de bloqueio [57] avalia o quantidade de tempo entre o FCP e o tempo de interação, que corresponde ao tempo em que o fluxo principal foi bloqueado por tempo suficiente para impedir que haja capacidade de resposta à introdução de dados por parte do utilizador. O fluxo principal considera-se bloqueado sempre que exista uma tarefa longa, isto é, uma tarefa com mais de 50 milissegundos. Diz-se que o fluxo principal foi bloqueado porque o browser não consegue interromper uma tarefa que se encontra em progresso. Assim, na eventualidade da interação do utilizador com a página durante a execução de uma tarefa longa, o browser terá que esperar que a tarefa termine antes de poder responder. Se a tarefa for longa o suficiente (> 50 ms), é provável que o utilizador note um atraso e se aperceba que a página é lenta ou inconstante. O tempo de bloqueio de cada tarefa longa, corresponde ao tempo excedente aos 50 ms estipulados para uma tarefa normal. Conclui-se assim que o tempo total de bloqueio da página se obtém através da soma de todos os tempos de bloqueio que ocorrem entre o FCP e o tempo de interação.

Considere-se o diagrama do fluxo principal do browser durante o carregamento de uma página, apresentado na figura 14.

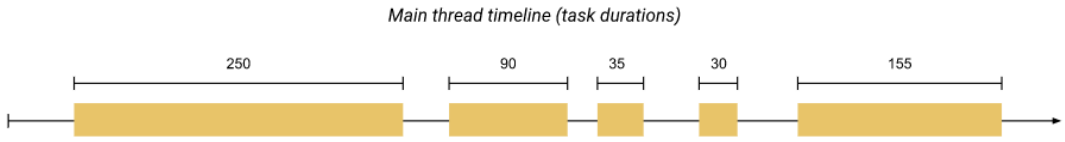


Figura 14 - Diagrama da Tarefa Principal de um browser durante o carregamento de uma página (retirado de [57])

O intervalo de tempo acima tem 5 cinco tarefas, 3 das quais são tarefas longas pois a sua duração excede os 50 milissegundos. O próximo diagrama identifica o tempo de bloqueio para cada uma das tarefas longas.

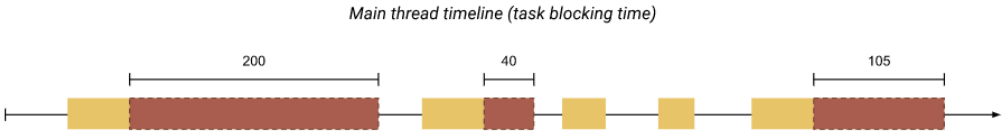


Figura 15 - Diagrama de tempo de bloqueio de cada tarefa longa[57]

Enquanto o tempo total gasto a executar tarefas é de 560 milissegundos, 345 milissegundos são considerados tempo de bloqueio.

	Task duration	Task blocking time
Task one	250 ms	200 ms
Task two	90 ms	40 ms
Task three	35 ms	0 ms
Task four	30 ms	0 ms
Task five	155 ms	105 ms
Total Blocking Time		**345 ms**

Figura 16 - Tempo Total de Bloqueio (retirado de [57])

O tempo total de bloqueio é uma métrica que se alia ao tempo de interação porque ajuda a quantificar a gravidade da inexistência de interatividade de uma página até esta se tornar comunicativa novamente. O tempo de interação considera uma página “interactivamente

confiável" se o fluxo principal estiver há pelo menos 5 segundos sem executar uma tarefa longa. Isto significa que 3 tarefas de 51 milissegundos distribuídas por 10 segundos podem atrasar o Tempo de Interação da mesma maneira que uma única tarefa de 10 segundos de duração. No entanto, estes dois cenários pareceriam muito diferentes a um utilizador. No primeiro caso, observava-se um tempo de bloqueio de 3 milissegundos, enquanto no segundo caso o tempo de bloqueio corresponderia a 9950 milissegundos. Quanto maior valor esta métrica apresentar, pior será a experiência do utilizador.

5.1.14 Tempo para o primeiro byte

Este intervalo de tempo corresponde ao tempo gasto para receber o primeiro byte da resposta a um pedido. É um dos indicadores chave do desempenho na Web. Algumas das formas de reduzir o tempo para o primeiro byte incluem a otimização do código, implementar cache ou renovar o hardware do servidor. De acordo com a Google [58], um bom valor para esta métrica deverá ser inferior a 200 milissegundos.

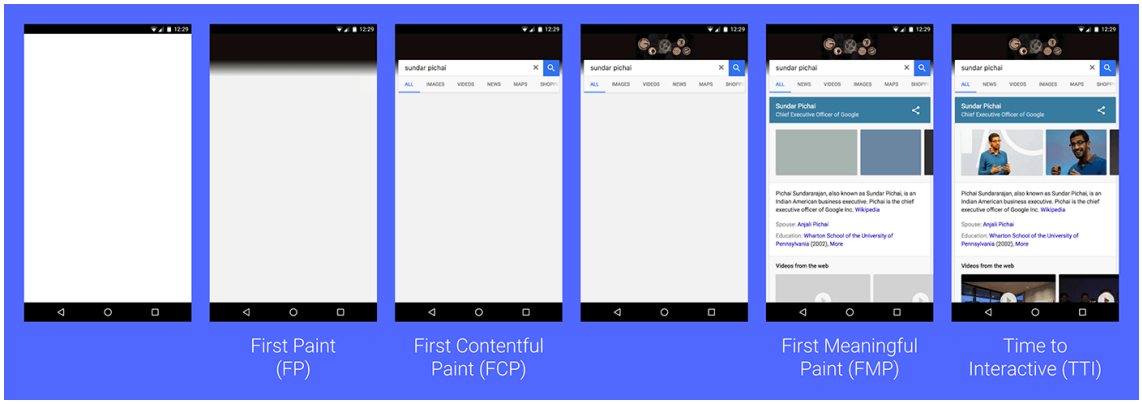


Figura 17 - Exemplo de carregamento de um página (retirado de [59])

Após o estudo destas métricas e correção dos problemas a elas associados, é suposto haver uma melhoria significativa nos resultados de cada uma, o que consequentemente significará uma melhoria considerável na performance da aplicação. Através da comparação de resultados (e.g. tempos) facilmente se comprova esta melhoria. Foi também efetuado um inquérito de satisfação com a equipa responsável pelo desenvolvimento da aplicação, uma vez que têm conhecimento do seu estado original.

5.2 Avaliação de metodologias

No presente subcapítulo serão apresentadas quais as metodologias utilizadas no desenvolvimento deste projeto, bem como o impacto causado pelas mesmas e respetivas justificações.

Após o período de pesquisa de tecnologias a usar na avaliação da aplicação e de metodologias que poderiam ser aplicadas na mesma, foi fornecido acesso ao código.

Inicialmente, foi efetuada uma revisão ao código usando um *software* denominado *Ndepend*. Com esta ferramenta é possível efetuar uma análise ao código e perceber onde se encontram falhas desde a estrutura e arquitetura do código até à própria qualidade deste.

Com esta análise foi possível aplicar uma das metodologias já apresentada neste documento, mais especificamente, a *Tree shaking.* Como descrito, esta metodologia tem como objetivo eliminar código que a aplicação compila, mas que não é utilizado em parte alguma. Uma das funcionalidades do *software* consiste em percorrer todo o código da aplicação e encontrar excertos de código não usados, como por exemplo métodos que não são chamados, ou condições que nunca se verificam. Estes métodos, denominados pelo Ndepend como "métodos mortos" são métodos que podem ser removidos do programa por nunca serem chamados e o relatório apresenta-os associados a um nível de profundidade. Um método morto cuja profundidade é 0 significa que nunca foi chamado, enquanto que um método que tenha profundidade 1 significa que é chamado por métodos que nunca são chamados, e assim sucessivamente.

Na análise efetuada observou-se um conjunto de potenciais fatores que poderiam ser ajustados para melhorar o funcionamento da aplicação.

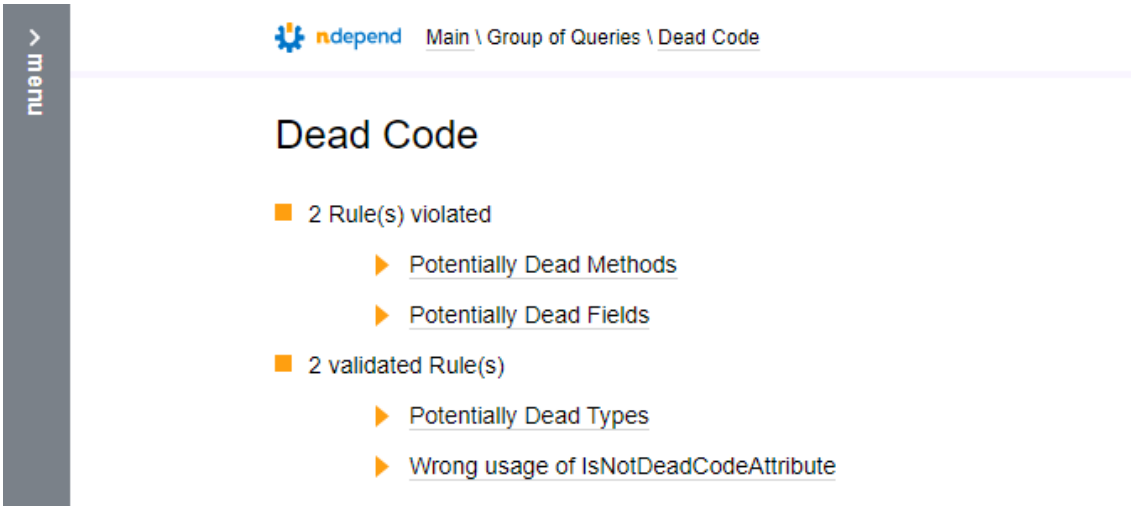


Figura 18 - Excerto do Relatório Ndepend – ATD

Na figura 18 apresenta-se uma parte dos resultados do Ndepend a um dos componentes da aplicação, o componente *Glintths.ATD.* O relatório indica a possibilidade de haver métodos,

campos ou tipos que nunca são utilizados, ou a utilização errada do atributo *IsNotDeadCode*. O *software* caracteriza estes problemas como regras a cumprir no código da aplicação e notifica o desenvolvedor quando não são cumpridas. Com esta análise foi possível verificar a existência de duas regras que não foram cumpridas, métodos e campos que não são chamados, e duas regras que foram validadas, uma vez que não foram encontrados tipos não chamados nem se verificou a utilização errada do atributo *IsNotDeadCode.* Este atributo é utilizado quando, por exemplo, um método é chamado dinamicamente e pode ser considerado, erradamente, como um método morto pela análise do Ndepend.

Na figura 19 apresentam-se problemas semelhantes no componente *Glintths.Patient*, onde se verifica o incumprimento das regras relativas a métodos e tipos mortos.

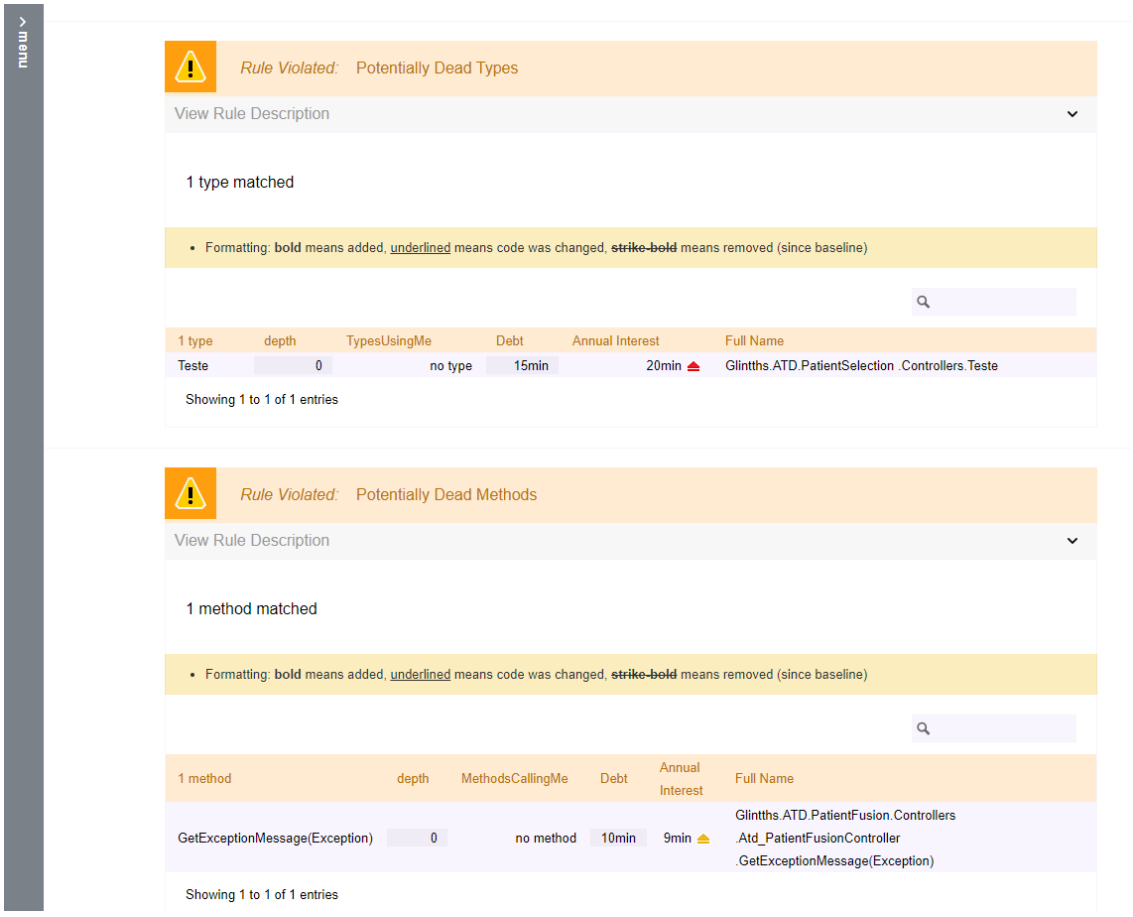
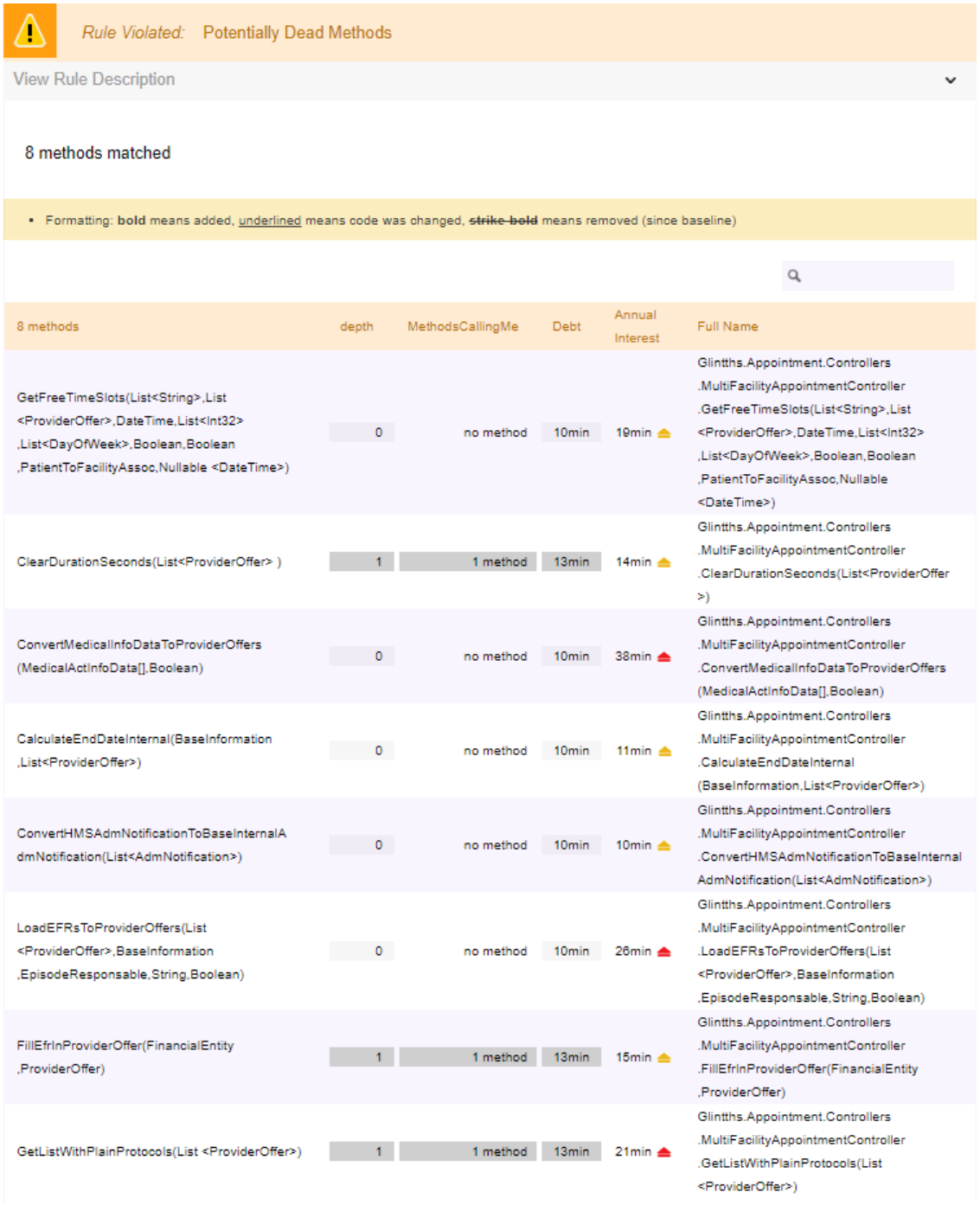


Figura 19 - Excerto do Relatório Ndepend - Patient

Por fim, na análise ao componente *Glintths.Appointment*, é possível observar a existência de um grande de número de métodos que não servem nenhum propósito, no entanto estão a ser compilados sem nunca serem executados.

Como se pode verificar na figura 20, alguns métodos como *GetFreeTimeSlots*, nunca são chamados, sendo que a sua utilidade é nula, e no caso dos métodos como *ClearDurationSeconds*,

cuja profundidade é 1, significa que são apenas chamados por métodos mortos, o que também não proporciona nenhuma funcionalidade para o programa.



Rule Violated: Potentially Dead Methods

View Rule Description

8 methods matched

- Formatting: **bold** means added, underlined means code was changed, ~~**strike bold**~~ means removed (since baseline)

8 methods	depth	MethodsCallingMe	Debt	Annual Interest	Full Name
GetFreeTimeSlots(List<String>,List <ProviderOffer>,DateTime,List<Int32> ,List<DayOfWeek>,Boolean,Boolean ,PatientToFacilityAssoc,Nullable <DateTime>)	0	no method	10min	19min	Glintths.Appointment.Controllers .MultiFacilityAppointmentController .GetFreeTimeSlots(List<String>,List <ProviderOffer>,DateTime,List<Int32> ,List<DayOfWeek>,Boolean,Boolean ,PatientToFacilityAssoc,Nullable <DateTime>)
ClearDurationSeconds(List<ProviderOffer>)	1	1 method	13min	14min	Glintths.Appointment.Controllers .MultiFacilityAppointmentController .ClearDurationSeconds(List<ProviderOffer >)
ConvertMedicalInfoDataToProviderOffers (MedicalActInfoData[],Boolean)	0	no method	10min	38min	Glintths.Appointment.Controllers .MultiFacilityAppointmentController .ConvertMedicalInfoDataToProviderOffers (MedicalActInfoData[],Boolean)
CalculateEndDateInternal(BaseInformation ,List<ProviderOffer>)	0	no method	10min	11min	Glintths.Appointment.Controllers .MultiFacilityAppointmentController .CalculateEndDateInternal (BaseInformation,List<ProviderOffer>)
ConvertHMSAdmNotificationToBaseInternalA dmNotification(List<AdmNotification>)	0	no method	10min	10min	Glintths.Appointment.Controllers .MultiFacilityAppointmentController .ConvertHMSAdmNotificationToBaseInternal AdmNotification(List<AdmNotification>)
LoadEFRsToProviderOffers(List <ProviderOffer>,BaseInformation ,EpisodeResponsable,String,Boolean)	0	no method	10min	26min	Glintths.Appointment.Controllers .MultiFacilityAppointmentController .LoadEFRsToProviderOffers(List <ProviderOffer>,BaseInformation ,EpisodeResponsable,String,Boolean)
FillEfrInProviderOffer(FinancialEntity ,ProviderOffer)	1	1 method	13min	15min	Glintths.Appointment.Controllers .MultiFacilityAppointmentController .FillEfrInProviderOffer(FinancialEntity ,ProviderOffer)
GetListWithPlainProtocols(List <ProviderOffer>)	1	1 method	13min	21min	Glintths.Appointment.Controllers .MultiFacilityAppointmentController .GetListWithPlainProtocols(List <ProviderOffer>)

Figura 20 - Excerto do relatório Ndepend - Appointment

Embora seja não seja possível determinar exatamente o impacto da correção destes problemas, pode-se admitir que a redução de código a ser analisado, compactado e executado tem uma influência positiva na performance da aplicação [60].

Os ficheiros e *scripts* enviados pela rede são geralmente comprimidos, o que significa que o seu tamanho após a descompressão aumenta significativamente. O *browser* lida com o tamanho real desses ficheiros, tendo de os analisar e compilar. Esta ação pode levar a um aumento no tempo de processamento das páginas e pode obrigar a um consumo extra de recursos da máquina. Para comparação, na figura 21, pode-se observar a diferença entre dois tipos de ficheiros com o mesmo tamanho quando são transferidos para um *browser*.

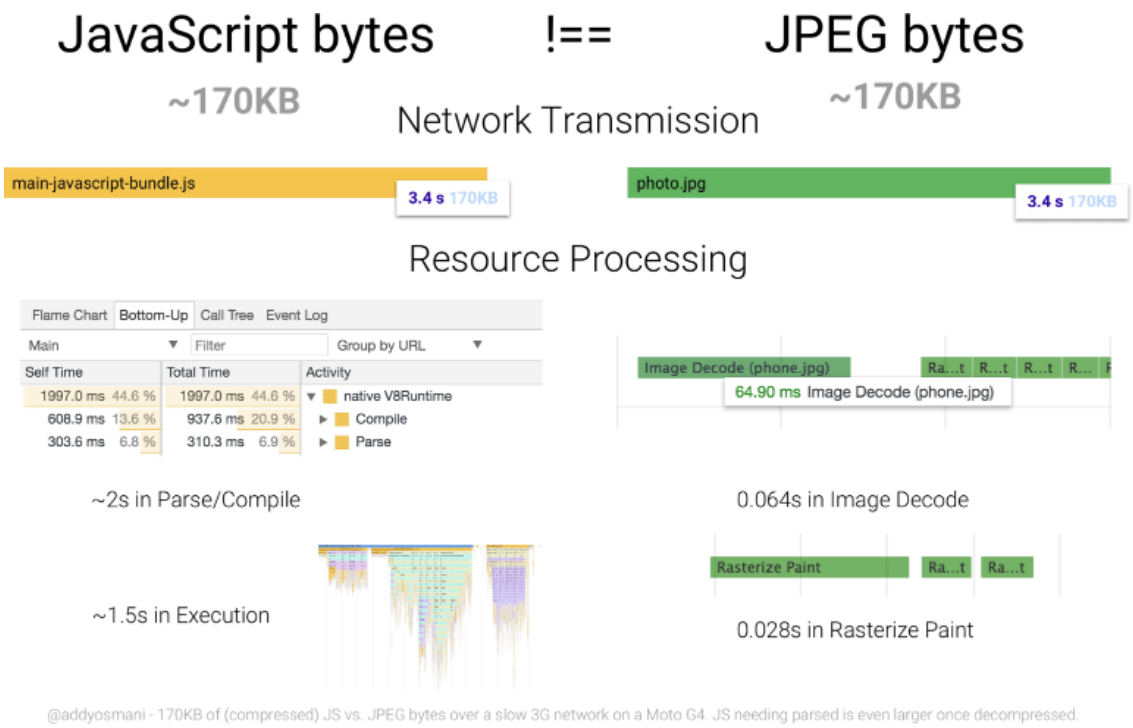


Figura 21 - Custo do processamento da analise e compilação de um ficheiro JavaScript e uma imagem JPEG (retirado de [61])

Embora ambos os ficheiros tenham o mesmo tamanho, o ficheiro JS necessita de passar por um processo demorado de análise, compilação e execução.

Prosseguiu-se à análise do código indicado pelo *Ndepend* e verificou-se que os métodos identificados não contribuíam para o funcionamento da aplicação, pelo que foram eliminados. Esta ação permitiu obter resultados positivos numa nova análise do *software*. Como se pode verificar na figura 22, as 4 regras analisadas pelo Ndepend foram cumpridas e garantiu-se que o programa não continha mais código morto, tendo sido aplicada a metodologia de *Tree Shaking*.

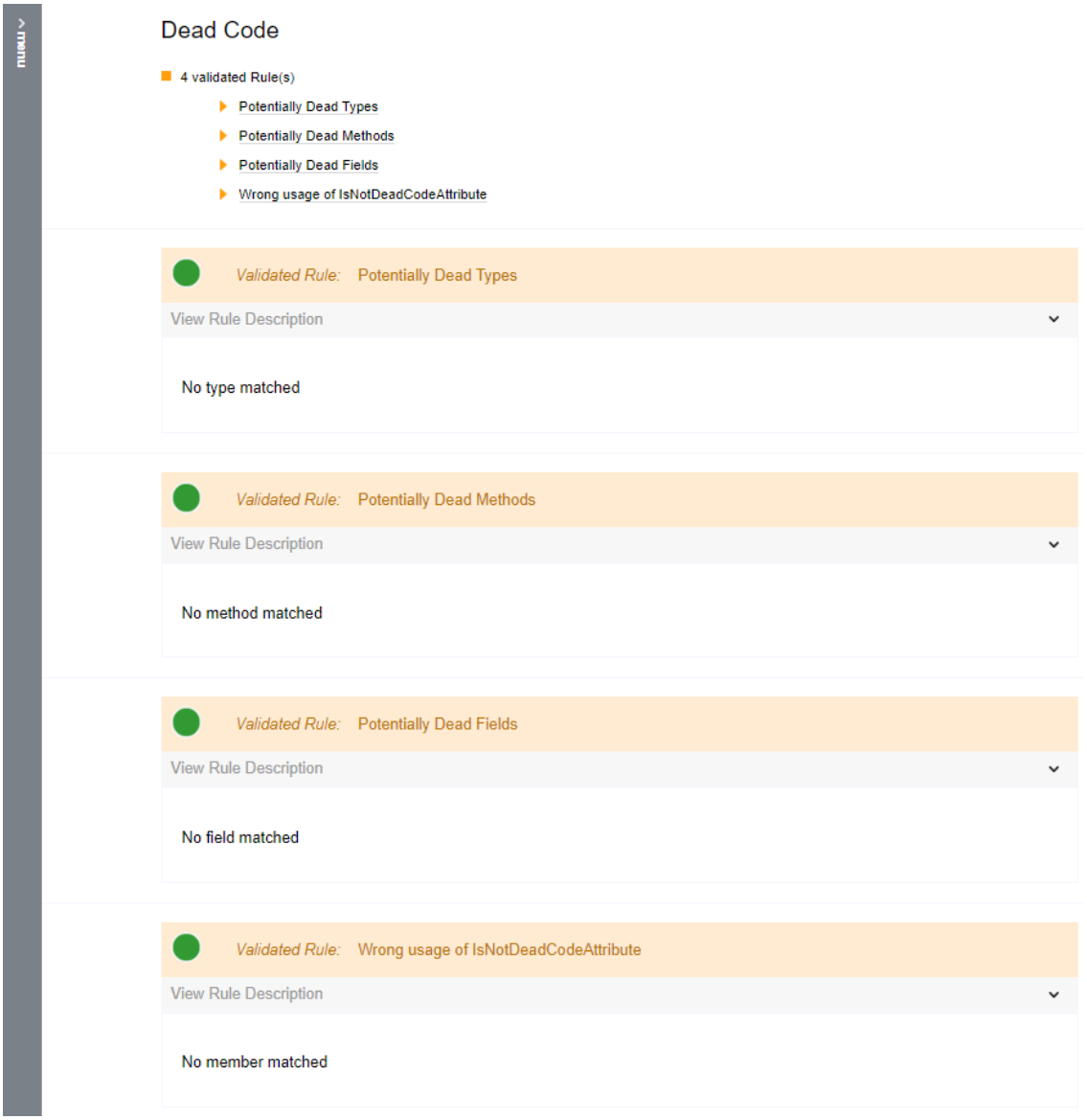


Figura 22 - Resultado da segunda análise do Ndepend aos 3 componentes

Uma vez concluído o passo de eliminação de código não usado, seguiu-se para identificação de possíveis casos para se aplicar a minificação.

A minificação é o processo de remoção de caracteres desnecessários do código sem alterar a funcionalidade do mesmo. Esses caracteres podem ser espaços em branco, quebras de linha, comentários ou até, em casos mais extremos, o encurtamento de nomes de variáveis ou de métodos. No entanto, a remoção destes caracteres torna o código, na maior parte das vezes, ilegível, como se pode verificar na figura 23.

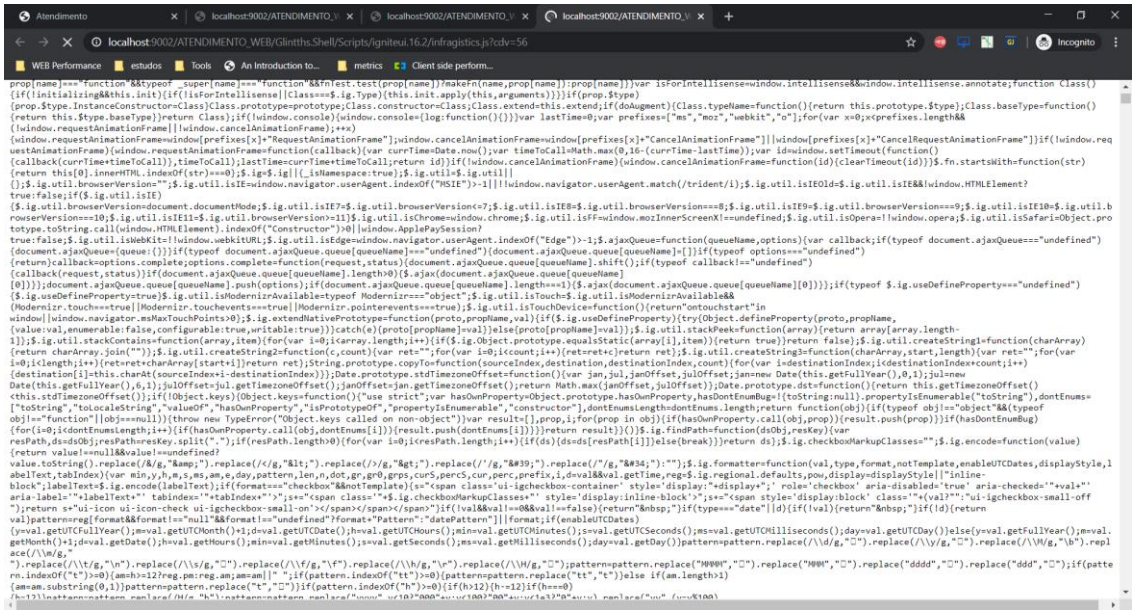


Figura 23 - Exemplo de um ficheiro do ATD minificado

Por essa razão, a sua aplicação deverá ocorrer somente em ficheiros predefinidos que não vão sofrer alterações. A título de exemplo, os ficheiros CSS utilizados pela empresa em todas as suas aplicações, e em que a versão enviada para o cliente contém os ficheiros minificados, ficando a versão normal com os desenvolvedores, para uma eventual atualização ou continuação do desenvolvimento no futuro.

No caso do Atendimento, que é uma aplicação ainda em desenvolvimento e com constantes atualizações, decidiu-se, em conjunto com pessoas responsáveis, que não se aplicaria a minificação por duas razões. Em primeiro lugar, os ficheiros cujo tamanho é consideravelmente grande, que raramente ou nunca são alterados e que seriam candidatos perfeitos para a aplicação desta metodologia já se encontram minificados, como se pode observar na figura 24, em que os ficheiros estão caracterizados com ".min" no seu nome.

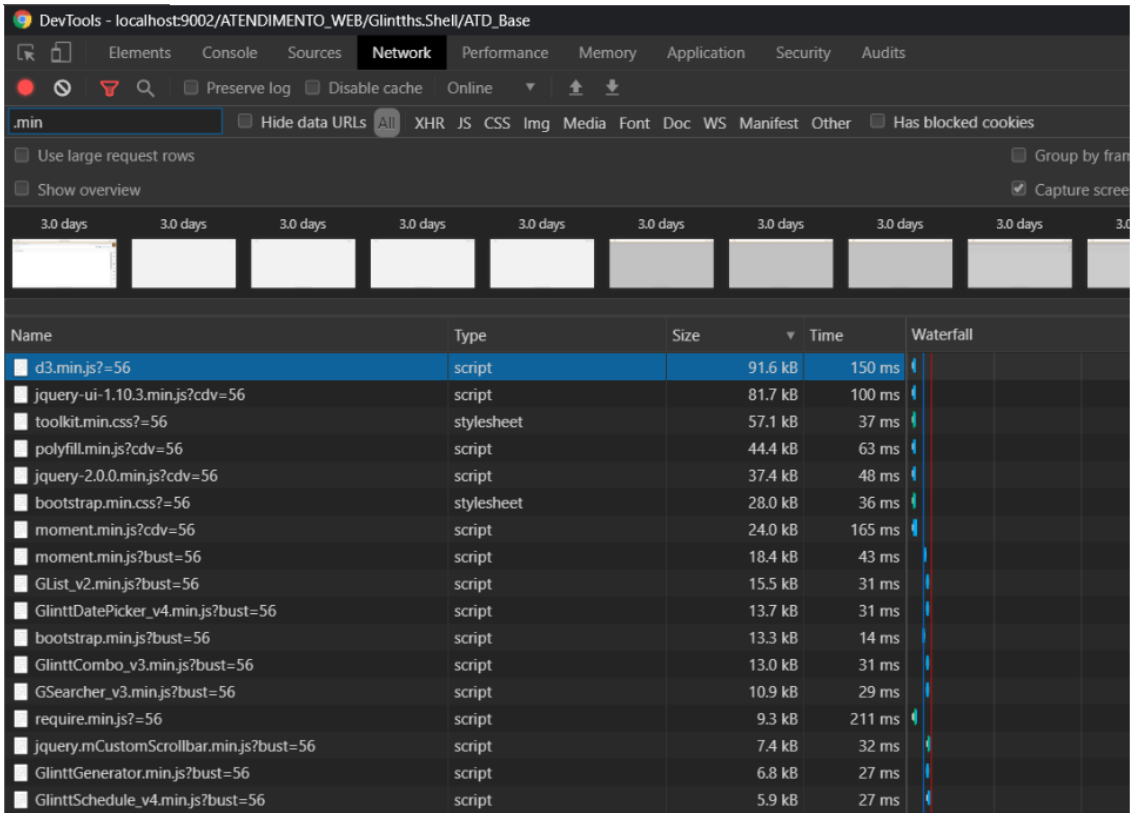


Figura 24 - Exemplos de alguns ficheiros do ATD minificados

Por outro lado, todos os ficheiros transferidos pela rede estão sujeitos a um *software* de compressão denominado *Gzip*. Este processo é responsável por comprimir ficheiros até 80 por cento do seu tamanho inicial, o que resulta em melhores tempos de carregamento de uma página [61].

O *browser* utiliza o *Gzip* para reduzir a quantidade total de informação transferida para o cliente.Na figura 25 está representado este processo, em que inicialmente o browser requisita um ficheiro ao servidor. Este envia uma versão comprimida do ficheiro requisitado, diminuindo o tamanho do ficheiro e tornando mais rápido o seu descarregamento. De seguida, o browser efetua a descompressão do ficheiro e compila a página.

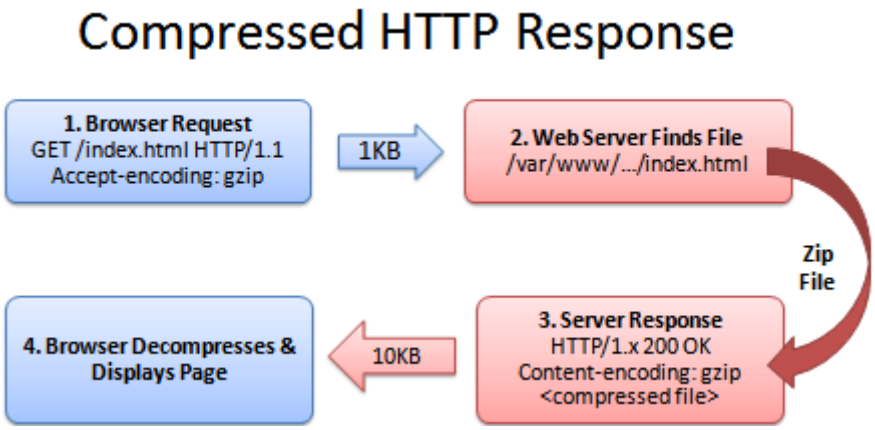


Figura 25 - Funcionamento do Gzip (retirado de [62])

Comparativamente com a minificação, a compressão de ficheiros atua de forma diferente, principalmente porque não existe perda de informação. No entanto, os seus resultados podem atingir valores bastante satisfatórios, como é o caso dos seguintes exemplos de ficheiros no ATD.

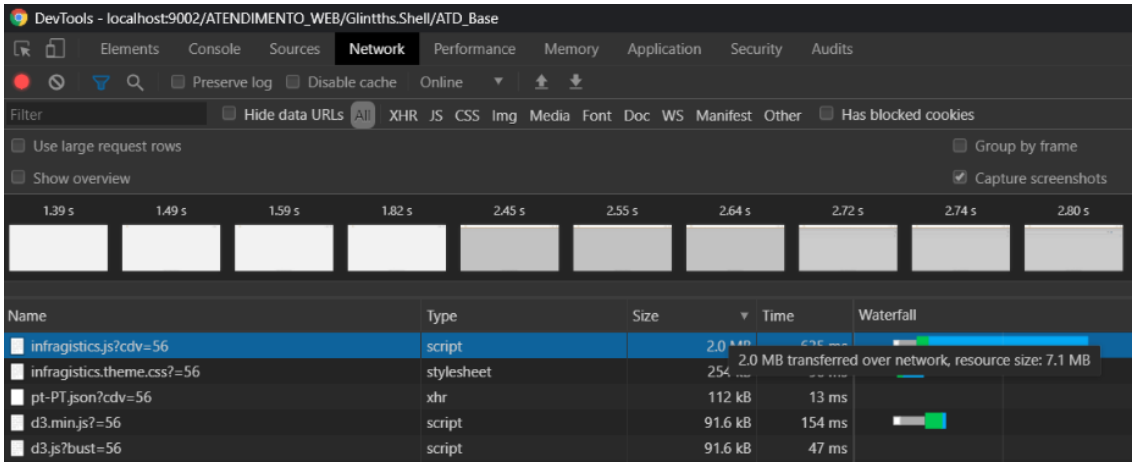


Figura 26 - Exemplo do efeito da compressão via Gzip no ATD

Na figura 26 podemos observar o ficheiro *infragistics.js,* cujo tamanho real é 7.1 MB, a ser transferido por 2 MB. Este valor corresponde aproximadamente a uma redução de 71 % do tamanho inicial do ficheiro.

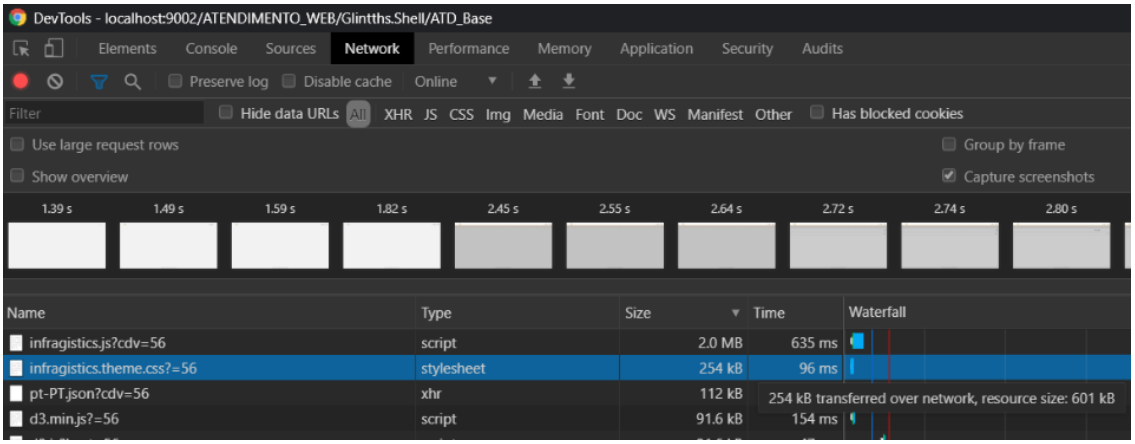


Figura 27 - Segundo exemplo da compressão via Gzip no ATD

Na figura 27, o ficheiro *infragistics.theme.css,* mesmo sendo um ficheiro muito menor que o *infragistics.js*, pode-se observar uma redução no tamanho do ficheiro em cerca de 58 %. Apesar de o *browser* ser obrigado a analisar, compilar e executar os ficheiros com o seu tamanho real, estes, ao serem transferidos com um tamanho bastante inferior, permitem uma diminuição no tempo associado à sua transferência.

Relativamente à metodologia *lazy loading*, é mais eficaz quando a informação a descarregar corresponde a um elevado número de imagens ou em listas demasiado grandes que podem nem ter espaço para aparecer no ecrã. Relativamente ao ATD, as imagens são praticamente inexistentes. Quanto às listas, o principal foco é na pesquisa de doentes. Ao pesquisar um nome, são apresentados diversos resultados que inundam o ecrã com informação. Para contrariar este problema, resolveu-se efetuar duas pesquisas iniciais. Na primeira pesquisa são apresentados os doentes que têm consultas naquele dia (dia em que a pesquisa é efetuada).

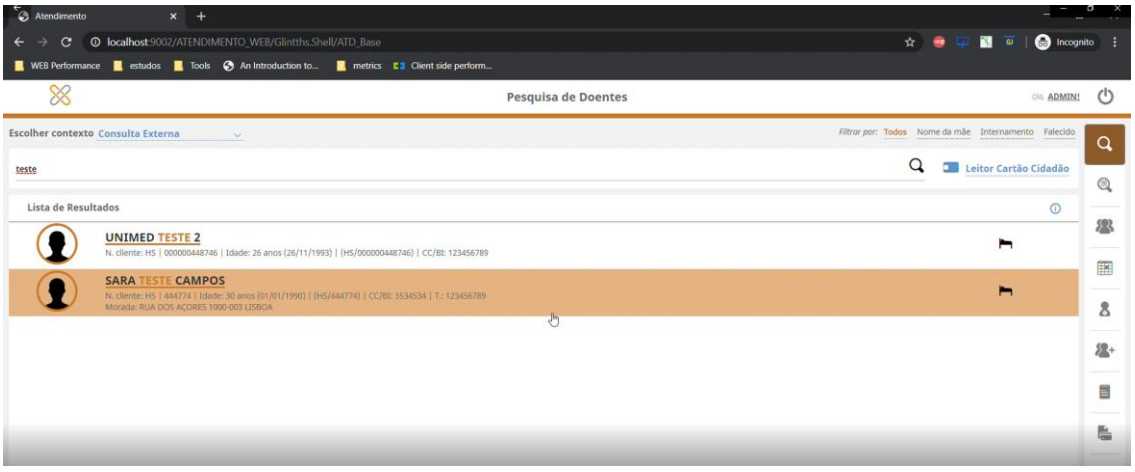


Figura 28 - Primeiros resultados da pesquisa de doentes

Na segunda pesquisa é apresentado um conjunto de resultados que, somados com os resultados da primeira, equivalem a 15 doentes. Desta forma, quando o utilizador procurar um doente que não esteja nos primeiros resultados, ao fazer *scroll* na lista e chegar ao fim desta,

iniciar-se-á uma nova pesquisa com mais 15 doentes. Assim evita-se um sobre carregamento de informação na página, limita-se o consumo de dados da rede e torna os pedidos ao servidor menores e mais rápidos. Uma vez que esta funcionalidade já se encontrava implementada e não existindo casos semelhantes nos outros componentes da aplicação, a metodologia não foi aplicada.

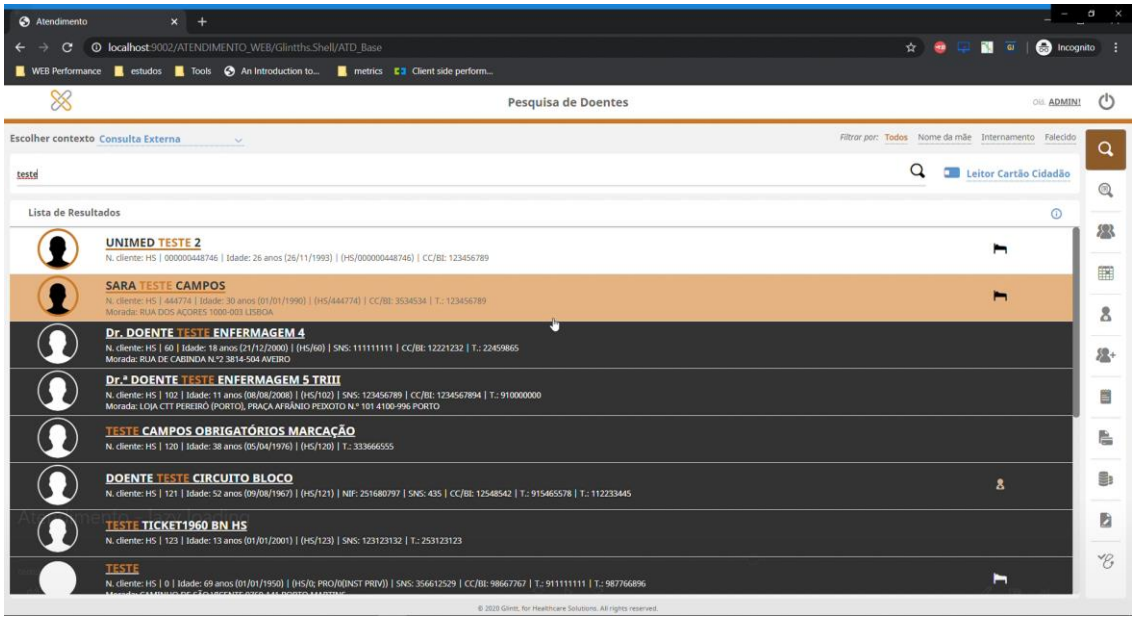


Figura 29 – Resultado completo da pesquisa de doentes

Depois da compreensão deste processo prosseguiu-se para a aplicação da metodologia de pré-procura complementada com as metodologias *preemptive start* e conclusão antecipada.

Pode-se afirmar que estas metodologias se complementam porque ao efetuar a pré-procura de conteúdo, o *browser* recebe informação de que existem recursos que tem grande probabilidade de serem usados no futuro, seja na página seguinte à atual ou na própria página. Assim sendo, estamos perante um caso de *preemptive start,* onde o utilizador ainda não sabe, mas já começou a "caminhada" para o próximo ponto da aplicação, tornando assim a "viagem" mais curta. De forma a não interferir com os tempos de carregamento da página, esta metodologia possui uma característica que garante que o *browser* só inicia a procura de recursos futuros nos momentos em que este se encontra inativo. Logo que o *browser* contenha toda a informação sobre os recursos descarregados em segundo plano, vai guardá-los em cache para que os possa apresentar assim que sejam requisitados.

A cache de um *browser* é um componente de *software* utilizado para guardar informação para que futuros pedidos possam ser atendidos de forma mais célere no futuro. Quando se visita um *website,* o *browser* recolhe pedaços de informação da página e guarda-os no computador[63].

Com a pré-procura é possível inverter a ordem do processo acima descrito, onde se pode guardar em cache a informação de uma página que ainda não foi visitada. No entanto, quando o utilizador escolhe visitar uma página em que algumas das suas propriedades foram

previamente descarregadas (*preemptive start*) estas serão apresentadas antes da página se encontrar totalmente construída, dando a ilusão de que a página já terminou o processo de carregamento. Desta forma, verifica-se um caso de conclusão antecipada a complementar a pré-procura.

Para que seja possível entender de forma mais clara, serão apresentados alguns exemplos práticos da aplicação destas metodologias no ATD.

A página inicial da aplicação, depois de efetuado o *login* com sucesso, apresenta o ecrã apresentado na figura 30.

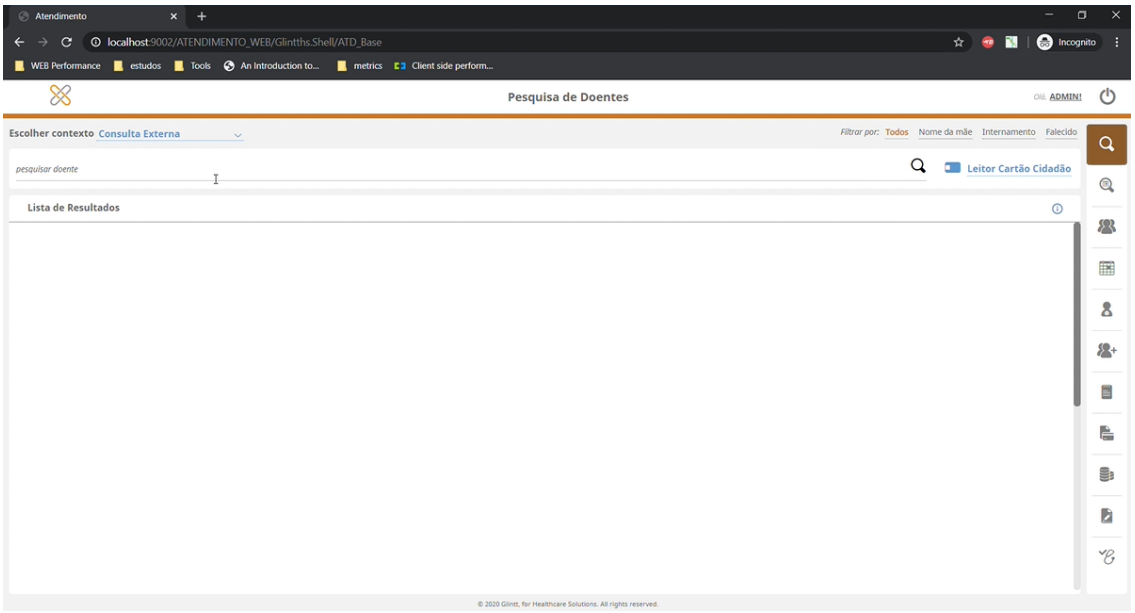


Figura 30 - Ecrã inicial da aplicação Atendimento

Antes de se aplicarem as metodologias, estes eram os resultados de alguns pedidos efetuados pela aplicação para abrir a página.

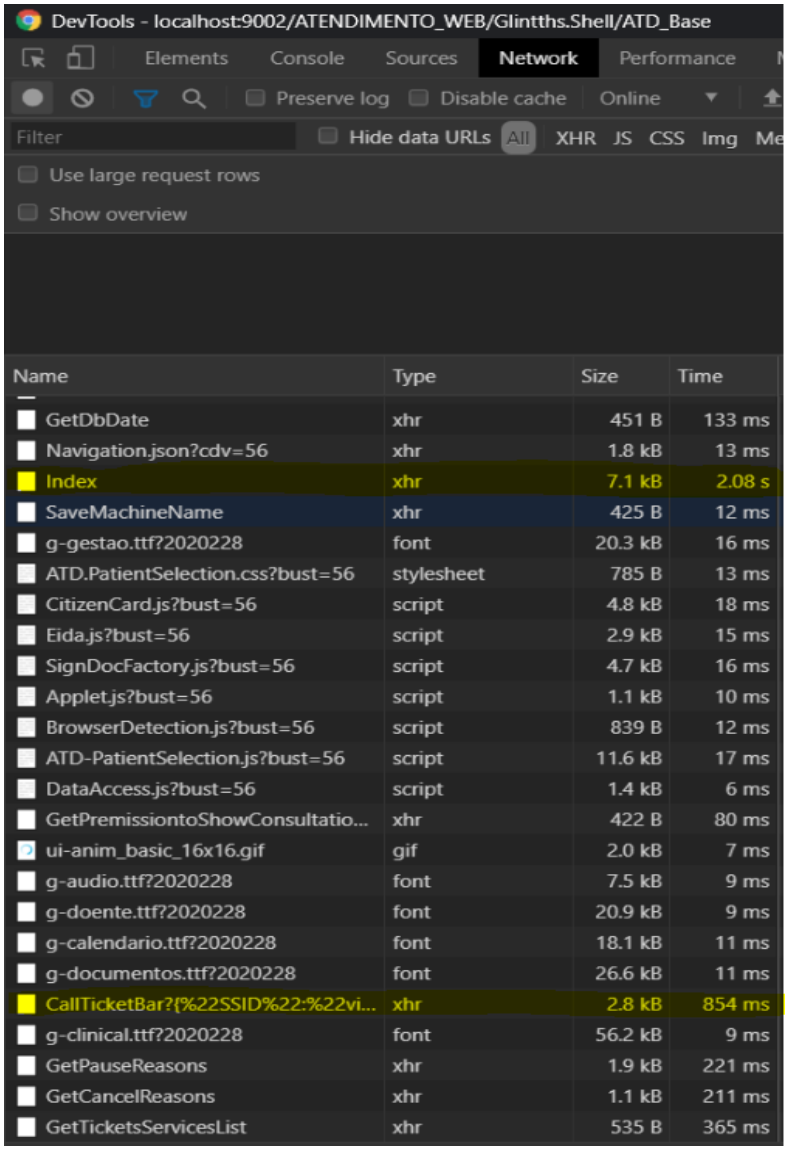


Figura 31 - Pedidos relativos à página inicial do ATD

Facilmente se identificam dois pedidos que sobressaem em relação aos restantes por terem o seu tempo de carregamento muito maior, *Index* e *CallTicketBar*. Ao observar-se este comportamento na página inicial, o *browser* foi instruído que começasse a descarregar estes recursos assim que o servidor enviasse uma resposta positiva em relação ao login. Desta forma, enquanto o *browser* constrói a página inicial e espera pela informação do servidor, vai descarregando os recursos que foram atribuídos inicialmente. No caso da página inicial, quando o *index* era chamado, o seu descarregamento já tinha começado previamente, o que tornava o processo mais curto. O resultado dessa ação mostra que, antes de se efetuar a pré-procura, o *index* demorava cerca de 2,08 segundos a ser descarregado e compilado. No entanto, como se pode verificar na figura 32, após se indicar ao *browser* que aquele recurso podia ser descarregado mais cedo, verificou-se uma melhoria no processo de cerca de 0,7 segundos. Este *index* corresponde à lista onde são apresentados os resultados da pesquisa de pacientes.

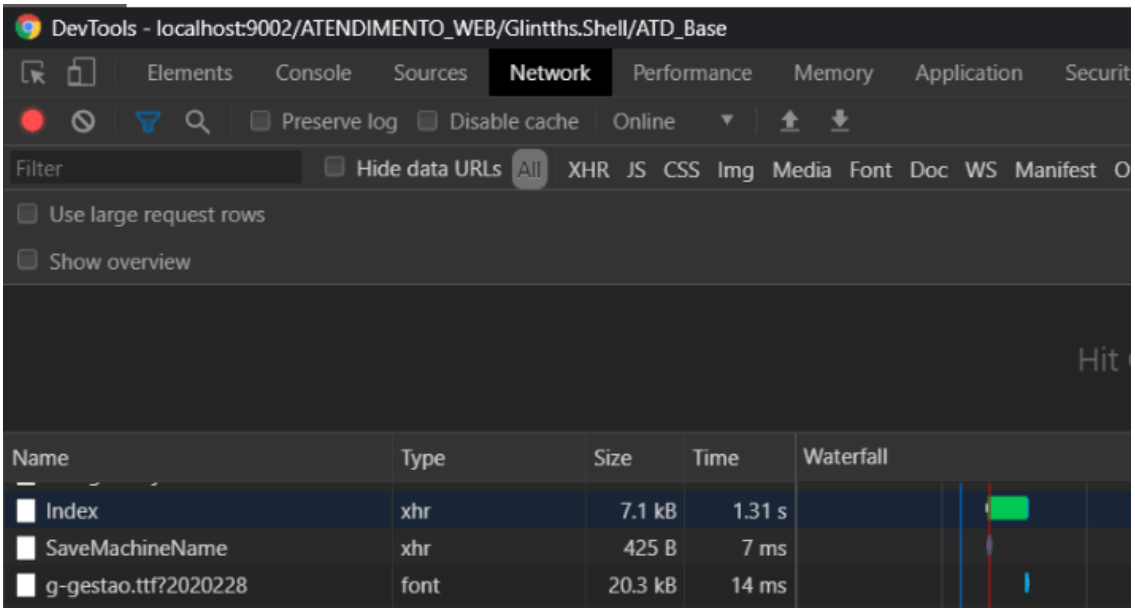


Figura 32 - Resultado da Pré-procura do *Index* da página inicial

Relativamente ao segundo pedido mais longo, este corresponde a uma barra lateral que não está visível logo na página inicial, no entanto causava um atraso notório no carregamento total da página. Assim sendo, informou-se o browser para que iniciasse o processo de carregamento da barra lateral assim que fosse possível. Desta forma, quando o pedido *CallTicketBar* é efetivamente chamado, já se encontra iniciado ou, num caso ótimo, totalmente descarregado e guardado em cache, pronto a ser apresentado. Assim foi possível obter um resultado positivo de aproximadamente 0,77 segundos, como se pode verificar na figura 33.

Figura 33 - Resultado da pré-procura do *CallTicketBar* na página inicial

Estes dois exemplos favorecem o tempo total de carregamento da página inicial do ATD que passou de 7,5 a 9 segundos para o 5,5 a 6,5 segundos. Todos estes valores e imagens apresentados anteriormente podem ser consultados no Anexo B.

Ao observar as vantagens que estas metodologias traziam à aplicação logo no arranque desta, decidiu-se aplicá-la noutros setores da mesma. Em conjunto com o orientador deste projeto na empresa, com conhecimento aprofundado sobre o funcionamento da aplicação, decidiu-se que os melhores componentes onde estas metodologias poderiam ser aplicadas seriam "Glintths.Appointment" (componente responsável pela marcação de consultas de um doente), "Glintths.ATD.Billing" (componente responsável por emitir a faturação de um cliente) e

“Glintths.Patient” (componente responsável pela ficha do doente). O critério que determinou a escolha destes 3 componentes, foi o facto de serem os mais utilizados pelo cliente.

Desta forma, iniciou-se a aplicação da pré-procura com a marcação de consultas. A marcação é um processo que pode ser acedido através do Cockpit do Doente (painel onde se pode consultar as sessões de um doente e algumas informações básicas) ou logo a partir do ecrã inicial. Quando se acedeu a este componente observou-se o seguinte comportamento dos pedidos da aplicação.

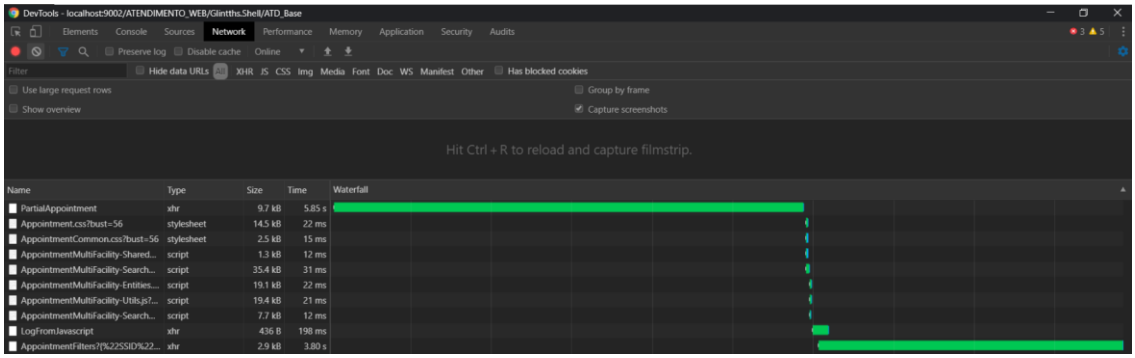


Figura 34 – Resultados correspondentes à Marcação de Consultas

Como é possível verificar na figura 34, existem dois momentos de elevada duração. O primeiro, *PartialAppointment,* corresponde à estrutura da página e o segundo, *AppointmentFilters,* corresponde a uma barra lateral onde são apresentados diversos filtros para a marcação de consultas como os dias, as horas e algumas preferências. Inicialmente, pretendia-se efetuar a pré-procura de ambos os recursos, no entanto, tal não foi possível porque o carregamento relativo ao *PartialAppointment* trazia demasiada informação, o que afetava o funcionamento normal da aplicação. Pelo contrário, os *AppointmentFilters* eram carregados sem nenhum tipo de problema pelo *browser*. Decidiu-se então efetuar a pré-procura dos mesmos no momento em que a página inicial estivesse completamente carregada. No Anexo B é possível encontrar a imagem completa relativa à página inicial (figura 75) e na parte inferior verifica-se que a chamada dos filtros da marcação ainda está pendente, porque a página inicial tinha acabado de ser carregada.

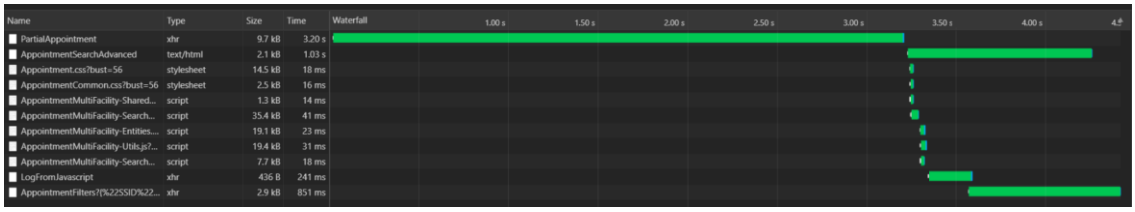


Figura 35 - Resultado da pré-procura dos filtros para marcação

Como se pode observar na figura 35, os resultados da pré-procura dos filtros para a marcação diminuíram drasticamente. Aproximadamente 3 segundos de diferença entre a figura 34 e a figura 35, o que representa valores significativos na resolução do problema da performance nesta aplicação.

É possível ainda observar um pedido na figura 35 que não se verifica na figura 34, isto porque quando se visita a página da marcação, há a possibilidade de abrir uma *combo box* que permite auxiliar o utilizador numa pesquisa avançada. Desta forma, efetuou-se uma pré-procura dessa caixa, que ocorre em simultâneo e em segundo plano relativamente aos outros recursos da página.

Figura 36 - Resultado da pesquisa avançada antes da pré-procura

Como corresponde a um elemento novo na página, o atraso na sua abertura é evidente. Desta forma, com a utilização da pré-procura é possível abrir a caixa de forma quase instantânea. Como se pode observar na figura 37, os resultados da pesquisa avançada diminuíram aproximadamente 0,79 segundos, em comparação com a figura 36.

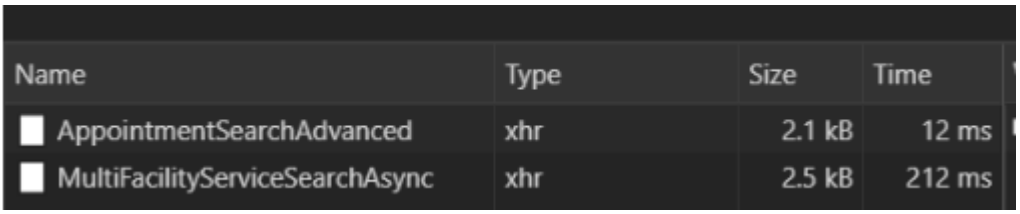


Figura 37 - Resultado da pesquisa avançada depois da pré-procura

Depois de uma intervenção com sucesso na Marcação, avançou-se para os componentes seguintes. Porém, tanto o componente do Paciente como o componente da Faturação têm de ser acedidos pelo ecrã “Cockpit do Doente”. Para atingir o ecrã referido é necessário pesquisar um doente no ecrã inicial como pode ser visto na figura 38.

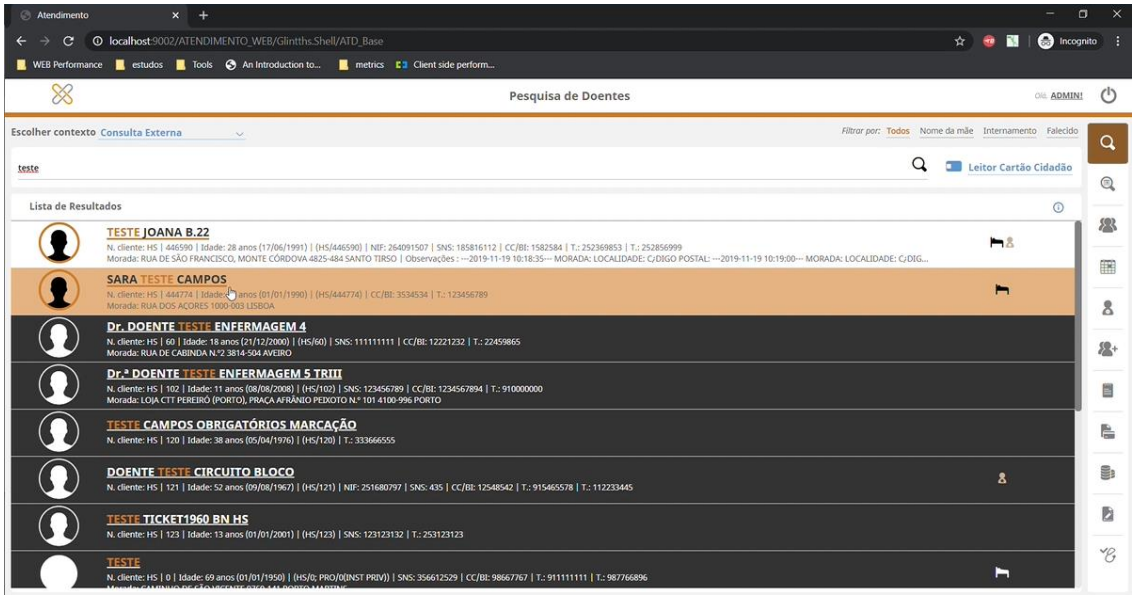


Figura 38 - Pesquisa de doentes

Quando o *browser* recebe a lista de doentes, inicia-se o processo de pré-procura do Cockpit do Doente, que é o ecrã apresentado quando o utilizador seleciona um doente. Desta forma, enquanto o utilizador procura o seu doente, o *browser* começa a preparação do ecrã seguinte. Inicialmente, o pedido correspondente ao ecrã em questão apresentava os valores da figura 39.

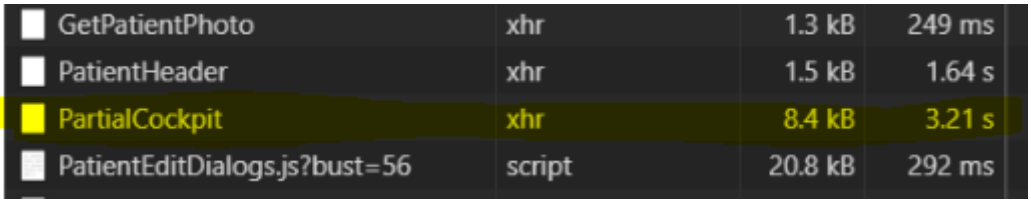


Figura 39 - Resultados do Cockpit do doente antes da pré-procura

Com a pré-procura a acontecer após a pesquisa de doentes, os resultados são aproximadamente 2 segundos mais rápidos.

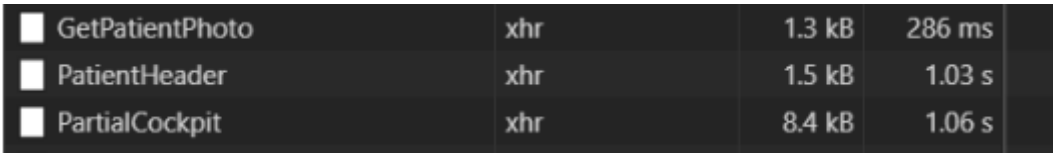


Figura 40 - Resultados do Cockpit do doente depois da pré-procura

Uma vez no Cockpit do Doente, o utilizador pode aceder a vários componentes, pelo que foi necessário encontrar uma forma de efetuar a pré-procura de alguns recursos sem sobrecarregar a rede.

Relativamente ao componente "Glintths.Patient", o principal foco residia na edição da ficha do doente por ser um processo muito utilizado pelo cliente. Assim, uma das principais páginas que

possui maior probabilidade de ser visitada é a da Ficha de Doente, pois é a partir dela que se pode efetuar a sua edição. A ficha de doente é acedida ao clicar no nome do doente no Cockpit.

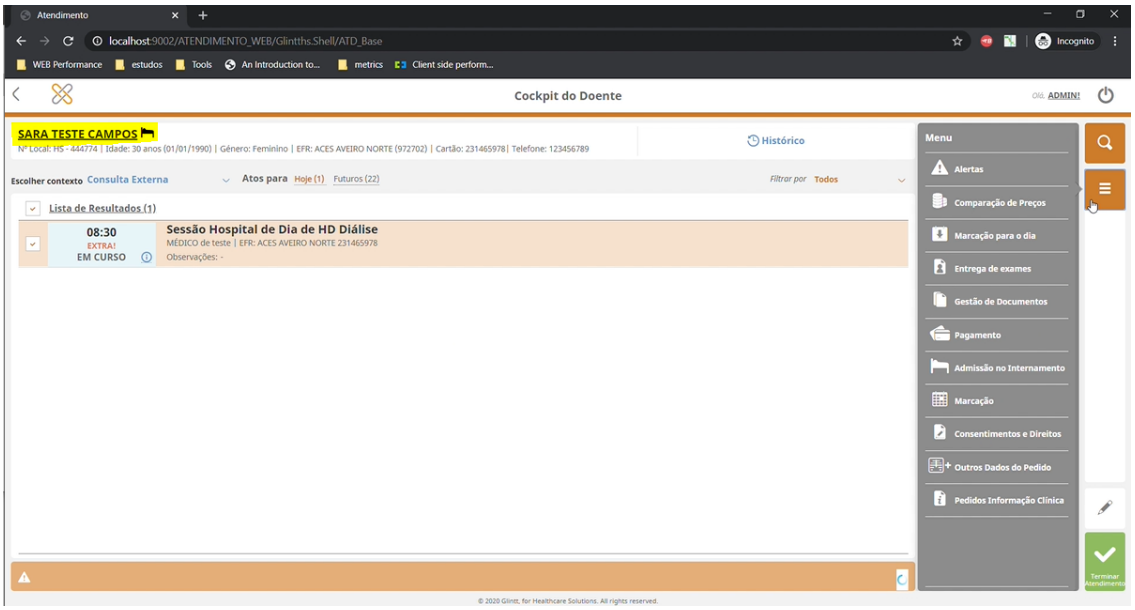


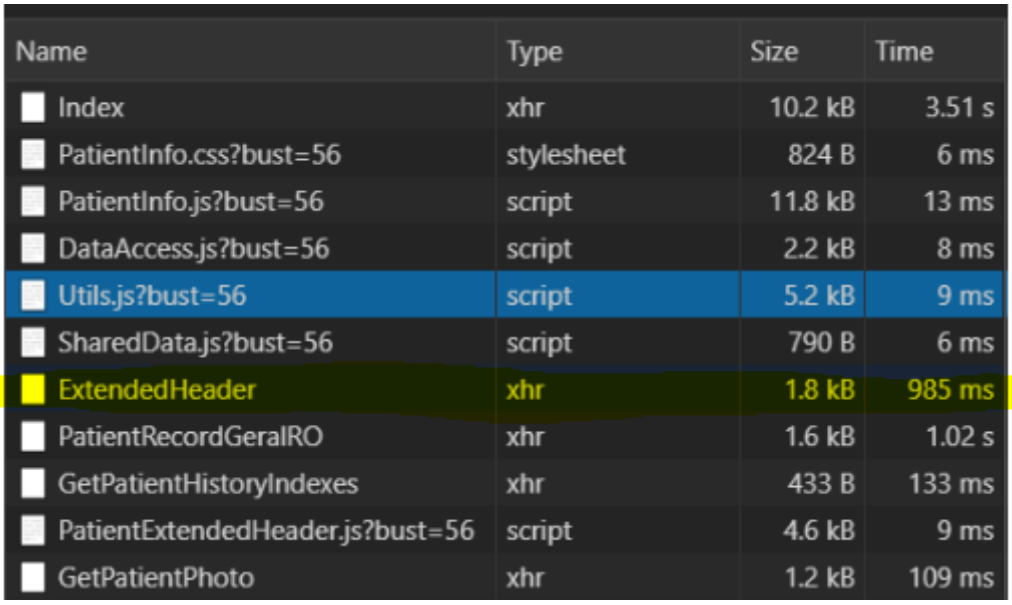
Figura 41 - Cockpit do Doente

Assim, decidiu-se que no Cockpit se faria uma pré-procura do cabeçalho da ficha do doente, uma vez que é um recurso leve mas que causava um ligeiro atraso na abertura da página seguinte, como se pode comprovar na figura 42.

Name	Type	Size	Time
Index	xhr	10.2 kB	3.89 s
PatientInfo.css?bust=56	stylesheet	822 B	7 ms
PatientInfo.js?bust=56	script	11.8 kB	13 ms
DataAccess.js?bust=56	script	2.2 kB	8 ms
Utils.js?bust=56	script	5.2 kB	10 ms
SharedData.js?bust=56	script	789 B	8 ms
ExtendedHeader	xhr	1.8 kB	1.77 s
PatientRecordGeralRO	xhr	1.6 kB	1.01 s
GetPatientHistoryIndexes	xhr	433 B	104 ms
PatientExtendedHeader.js?bust=56	script	4.6 kB	9 ms
GetPatientPhoto	xhr	1.2 kB	104 ms

Figura 42 - Resultado do cabeçalho da ficha de doente antes da pré-procura

Quando se aplicou a pré-procura no cockpit, observou-se uma descida de aproximadamente 0,8 segundos na construção do cabeçalho.



Name	Type	Size	Time
Index	xhr	10.2 kB	3.51 s
PatientInfo.css?bust=56	stylesheet	824 B	6 ms
PatientInfo.js?bust=56	script	11.8 kB	13 ms
DataAccess.js?bust=56	script	2.2 kB	8 ms
Utils.js?bust=56	script	5.2 kB	9 ms
SharedData.js?bust=56	script	790 B	6 ms
ExtendedHeader	xhr	1.8 kB	985 ms
PatientRecordGeralRO	xhr	1.6 kB	1.02 s
GetPatientHistoryIndexes	xhr	433 B	133 ms
PatientExtendedHeader.js?bust=56	script	4.6 kB	9 ms
GetPatientPhoto	xhr	1.2 kB	109 ms

Figura 43 - Resultado do cabeçalho da ficha de doente depois da pré-procura

Na ficha do doente é possível observar os dados pessoais do doente selecionado e no canto inferior direito existe um ícone de um lápis, sugerindo a opção de editar os seus dados.

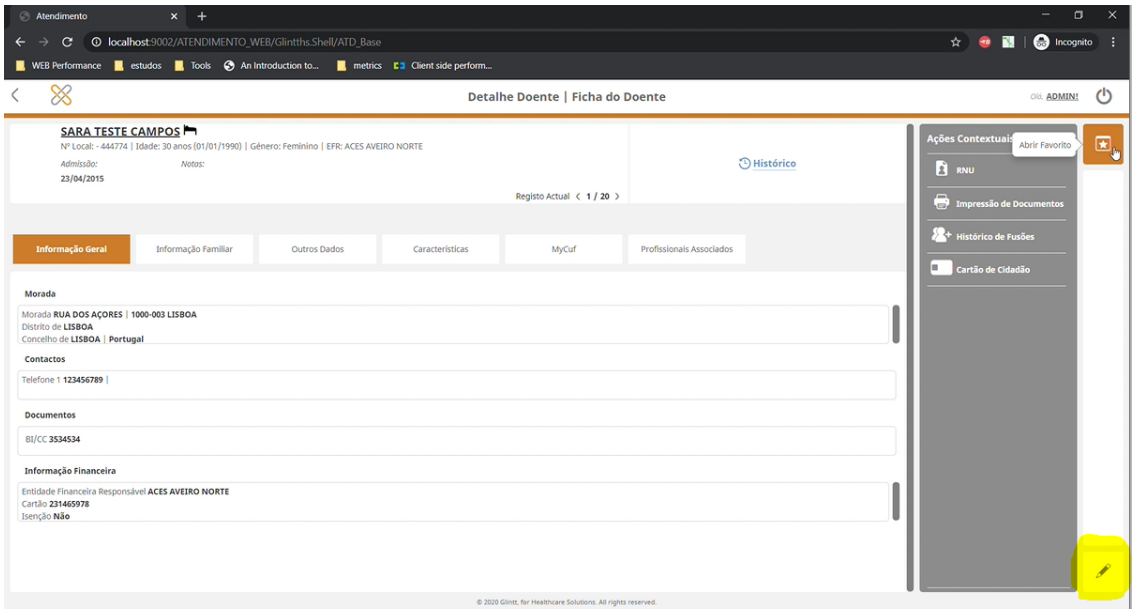


Figura 44 - Ficha do doente

Uma vez que o botão para editar a ficha do doente é clicado, o *browser* recebe um pedido para construção de um ecrã que se prolonga por um período exageradamente grande, podendo atingir os 10 segundos de espera, como se verifica no caso da figura 45.

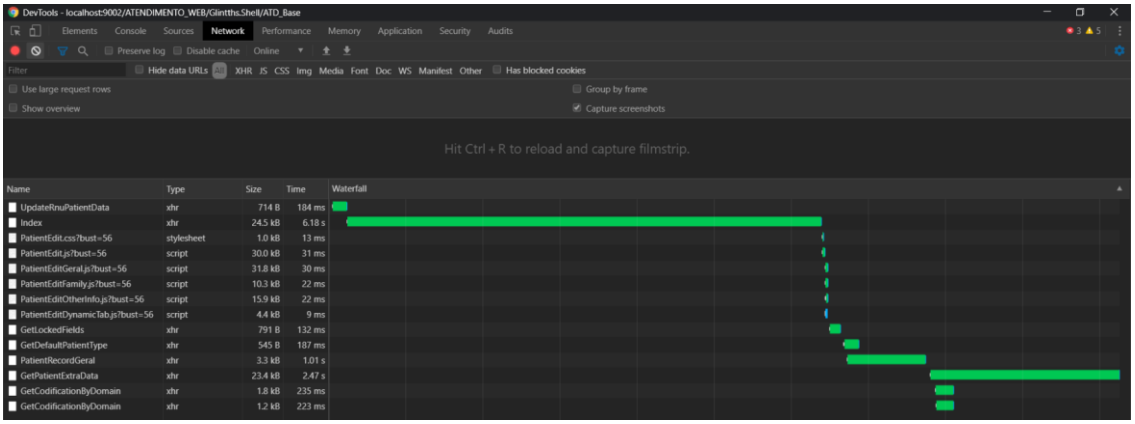


Figura 45 - Resultados da edição da ficha de doente antes da pré-procura

Como tal, decidiu-se aplicar a pré-procura do recurso principal da página, *o Index*. Esta ação acabou por permitir que o descarregamento e compilação da página de edição de um doente se iniciasse previamente para que, quando o utilizador clicasse na opção para editar, esta já se encontrasse parcialmente ou, num caso ótimo, totalmente preparada a ser apresentada. Usando esta metodologia, deixou-se de se necessitar de cerca de 10 segundos para abrir a página para passar a necessitar-se somente cerca de 4,3 segundos, como se pode verificar na seguinte 46.

Name	Type	Size	Time
UpdateRnuPatientData	xhr	714 B	192 ms
Index	xhr	24.5 kB	2.02 s
PatientEdit.css?bust=56	stylesheet	1.0 kB	6 ms
PatientEdit.js?bust=56	script	30.0 kB	21 ms
PatientEditGeral.js?bust=56	script	31.8 kB	27 ms
PatientEditFamily.js?bust=56	script	10.3 kB	14 ms
PatientEditOtherInfo.js?bust=56	script	15.9 kB	33 ms
PatientEditDynamicTab.js?bust=56	script	4.4 kB	7 ms
GetLockedFields	xhr	791 B	132 ms
GetDefaultPatientType	xhr	545 B	164 ms
PatientRecordGeral	xhr	3.3 kB	131 ms
GetPatientExtraData	xhr	23.4 kB	1.78 s
GetCodificationByDomain	xhr	1.8 kB	129 ms
GetCodificationByDomain	xhr	1.2 kB	117 ms

Figura 46 - Resultados da edição da ficha de doente após pré-procura

Uma vez concluídas com sucesso as modificações ao componente “Glintths.Patient”, seguiu-se para o próximo componente, o da faturação: “Glintths.ATD.Billing”. Para aceder a este componente também é necessário visitar o ecrã relativo ao Cockpit do Doente, pelo que se

efetuou outra pré-procura nesta página. Para a abertura da página relativa à faturação, o utilizador terá de procurar pela aba “Pagamento” na barra lateral do cockpit. Assim que o utilizador clicasse, antes de existir pré-procura, o *browser* apresentava os valores da figura 47.

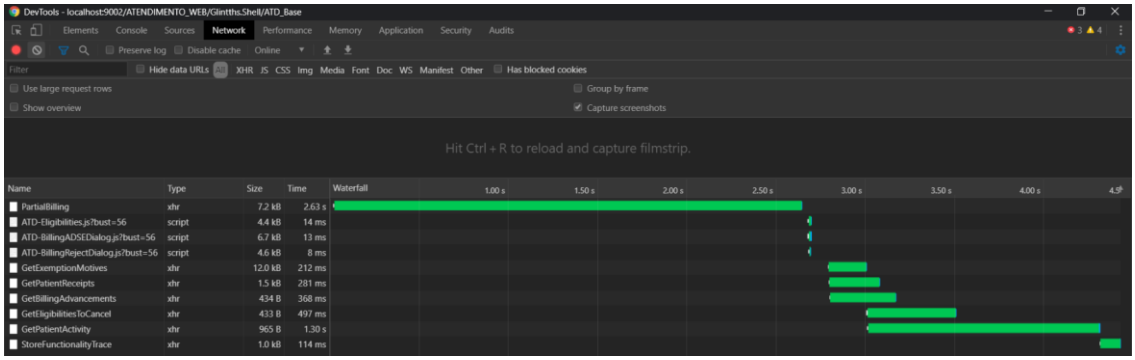


Figura 47 - Resultados do Pagmento antes da pré-procura

O pedido apresentado como *PartialBilling* corresponde à estrutura do ecrã de pagamento, logo, foi efetuada uma pré-procura nesse mesmo recurso para que a estrutura da página seguinte seja apresentada o quanto antes, no caso de ser selecionada. Desta forma, foi possível obter uma diferença de quase 2 segundos na abertura do mesmo ecrã.

Name	Type	Size	Time
PartialBilling	xhr	7.2 kB	163 ms
PendingDocuments	text/html	2.4 kB	1.95 s
Advances	text/html	4.3 kB	3.32 s
ATD-Eligibilities.js?bust=56	script	4.4 kB	14 ms
ATD-BillingADSEDialog.js?bust=56	script	6.7 kB	17 ms
ATD-BillingRejectDialog.js?bust=56	script	4.6 kB	9 ms
GetExemptionMotives	xhr	12.0 kB	157 ms
GetPatientReceipts	xhr	1.5 kB	298 ms
GetBillingAdvancements	xhr	434 B	324 ms
GetEligibilitiesToCancel	xhr	433 B	162 ms
GetPatientActivity	xhr	1.2 kB	229 ms
StoreFunctionalityTrace	xhr	1.0 kB	113 ms

Figura 48 - Resultados do Pagamento depois da pré-procura

As vantagens que estas metodologias produziram no *front end* desta aplicação são visíveis, pelo que foram ainda efetuadas outras alterações utilizando o mesmo método em diversas partes da aplicação embora com menor impacto. Isso pode ser visto no Anexo B.

As metodologias de pré-carregamento e pré-renderização não foram aplicadas no desenvolvimento deste projeto. A pré-renderização, sendo uma metodologia aconselhada para

páginas completas, como referido anteriormente, não se justificava o uso no Atendimento, que é uma *Single Page Application* [64] onde apenas certos aspetos da página são alterados e modificados na transição entre componentes, sendo o consumo de recursos mais limitado. O pré-carregamento é uma metodologia que não se encontra disponível no navegador *Internet Explorer*, característico dos utilizadores da aplicação, logo foi descartada a sua utilização. Na figura 49 estão representados os navegadores e respetivas versões que suportam o pré-carregamento.

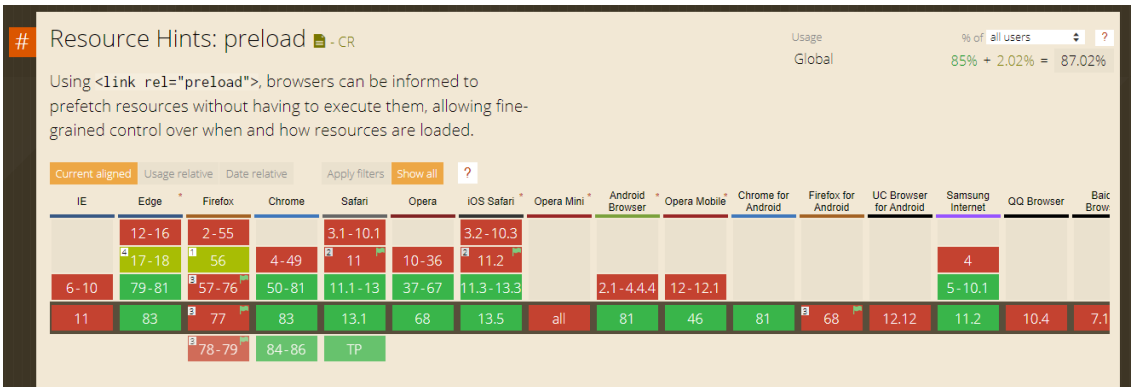


Figura 49 - Lista de navegadores e respetivas versões onde o pré-carregamento está disponível (retirado de [64])

5.3 Avaliação de tecnologias

Neste subcapítulo encontra-se detalhada a avaliação por parte das tecnologias utilizadas no desenvolvimento deste projeto.

Inicialmente, pretendia-se efetuar uma avaliação da performance do Atendimento através do *software* PageSpeed Insights, desenvolvido pela Google para avaliar o desempenho de uma página, além de oferecer sugestões sobre o aprimorar os resultados. No entanto, como a aplicação estava a ser testada localmente, o próprio programa indicava a utilização do *Lighthouse*, que é outro *software* de análise ao desempenho de aplicações desenvolvido pela Google e que se encontra nas DevTools do Chrome. Os primeiros resultados da análise ao Atendimento por parte do Lighthouse são apresentados na figura 50.

Figura 50 - Resultados da avaliação inicial do Lighthouse

Para compreender melhor os resultados da figura 50, é necessário compreender o sistema de pontuação do Lighthouse. A pontuação final da performance divide-se em três blocos, perfazendo um total de 100 pontos possíveis. O primeiro bloco varia entre os valores 0 e 49, caracterizados pela cor vermelha, o que significa que a performance se encontra num estado fraco e pouco otimizada. No segundo bloco, que percorre os valores entre 50 e 89 verifica-se um estado medíocre no desempenho da página, transmitindo a ideia de que há possibilidade para melhorar. Por fim, entre os 90 e os 100 pontos está a zona de melhor estado da performance, onde a maior parte dos critérios de avaliação foram cumpridos.

Como se pode verificar na figura 50, a aplicação apresenta um resultado final de 83, o que significa que pode haver melhorias, estabelecendo-se assim o objetivo de colocar o Atendimento na zona verde.

A avaliação por parte deste *software* realizou-se quando este se encontrava na sua versão 5.0, que desde então já sofreu uma atualização e, à data da escrita deste documento, se encontra na versão 6.0 [65].

Com a última versão disponibilizada pela Google, os critérios foram alterados, pelo que serão apresentados na figura 51 os critérios da versão 5.0.

Lighthouse 5

Audit	Weight
First Contentful Paint	20%
Speed Index	27%
First Meaningful Paint	7%
Time to Interactive	33%
First CPU Idle	13%

Figura 51 - Sistema de Pontuação Lighthouse 5.0 [66]

Relativamente ao *First Contentful Paint,* o Lighthouse estabeleceu os seguintes critérios [67]:

- De 0 a 2 segundos – zona verde
- De 2,1 a 4 segundos – zona laranja
- Acima de 4 segundos – zona vermelha

Relativamente ao *Speed Index,* foram estabelecidos os seguintes critérios [68]:

- De 0 a 4,3 segundos - zona verde
- De 4,4 a 5,8 segundos – zona laranja
- Acima de 5,8 segundos – zona vermelha

Relativamente ao *First Meanigful Paint,* foram estabelecidos os seguintes critérios [69]:

- De 0 a 2 segundos – zona verde
- De 2,1 a 4 segundos – zona laranja
- Acima de 4 segundos – zona vermelha

Relativamente ao *Time to Interactive,* foram estabelecidos os seguintes critérios [70]:

- De 0 a 3,8 segundos – zona verde
- De 3,9 a 7,3 segundos – zona laranja
- Acima de 7,3 segundos – zona vermelha

Relativamente ao *First CPU Idle,* não foi considerada por ser muito similar ao *Time to Interactive,* razão pela qual foi descontinuada na versão 6.0. [71]

Por último, a métrica apresentada na parte inferior da imagem 45, que corresponde ao tempo para o primeiro *byte*, foi considerada nesta avaliação. O lighthouse caracteriza esta métrica entre os seguintes valores [72]:

- De 0 a 0,1 segundos – zona verde
- De 0,11 a 0,2 segundos – zona laranja
- De 0,21 a 0,6 segundos– zona vermelha
- Acima de 0,6 segundos – O lighthouse assume erro na procura

Com esta informação, é possível avaliar agora os resultados do desempenho do Atendimento após a intervenções das metodologias descritas.

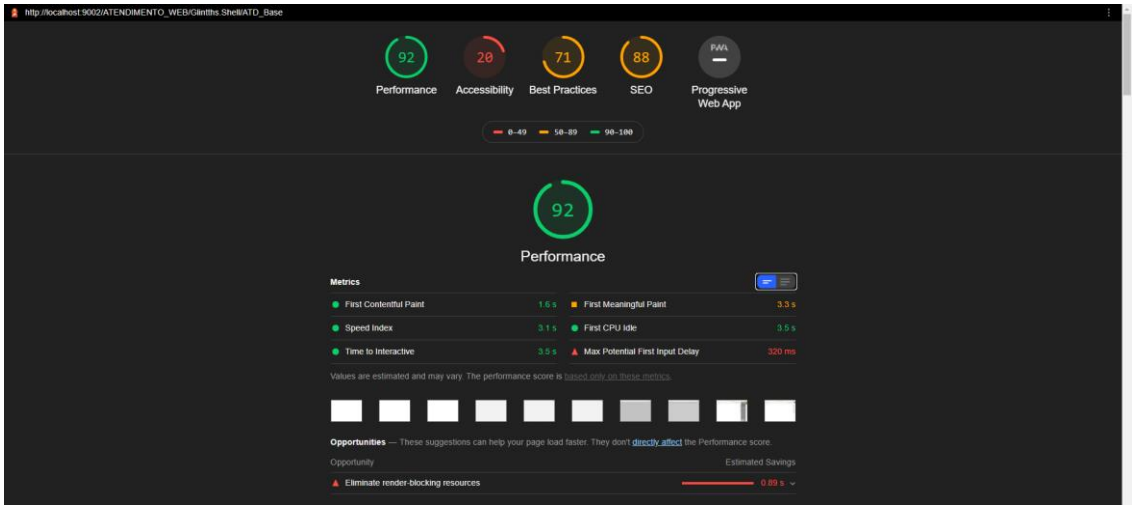


Figura 52 - Resultados da avaliação final do Lighthouse

Pela análise dos resultados ilustrados na figura 52, verifica-se o sucesso das intervenções na aplicação. Com uma pontuação de 92, foi possível colocar o Atendimento na zona verde da avaliação de desempenho, bem como 4 das 5 métricas consideradas.

A tecnologia Taurus permitiu a obtenção de resultados em larga escala, uma vez que possibilitava o teste da aplicação com vários utilizadores de forma contínua. Este *software* permitiu testar a aplicação através da criação de utilizadores virtuais e concorrentes criando relatórios compatíveis com a interface gráfica do Blazemeter, uma vez que foram desenvolvidos pela mesma empresa.

De modo a não sufocar a máquina durante a execução dos testes, estabeleceu-se um limite de 20 utilizadores simultâneos a aceder ao Atendimento durante 10 minutos. Estes utilizadores simulavam a abertura da aplicação, desde o *login* até à construção completa da página de pesquisa de doentes. Este teste permitiu obter aproximadamente 900 resultados, apresentados na figura 53.

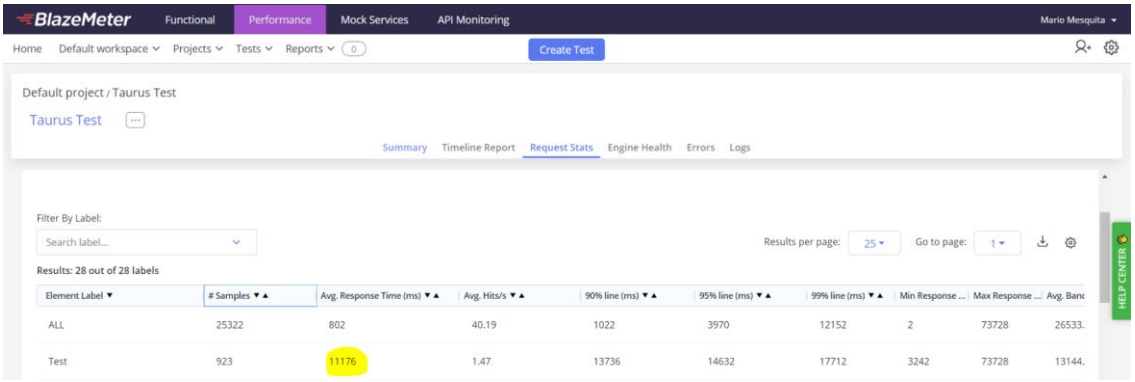


Figura 53 - Resultados iniciais da abertura da aplicação no Taurus

Na figura 53 é possível verificar que o tempo médio de resposta que cada utilizador obteve, desde a escrita do nome de utilizador e palavra passe, até à conclusão da página de pesquisa de doentes. O resultado de 11,176 segundos é cerca de 1,45 segundos superior ao resultado de um teste semelhante, efetuado após a aplicação de diversas metodologias de otimização, como se pode verificar na figura 54.

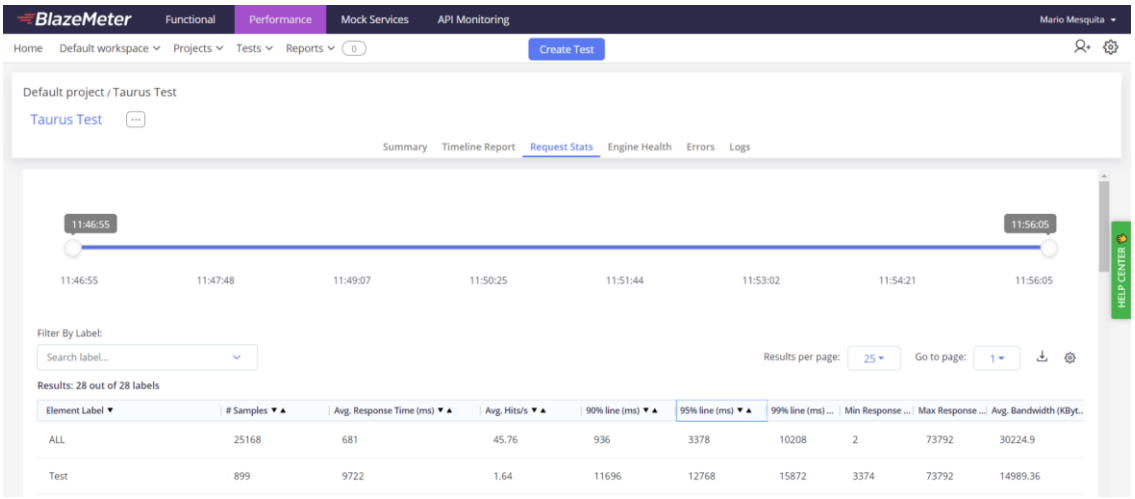


Figura 54 - Resultados finais da abertura da aplicação no Taurus

Por último, de forma a existir um termo de comparação, efetuaram-se testes semelhantes com 21 utilizadores concorrentes no Jmeter, para observar a avaliação do comportamento da aplicação por outro *software*.

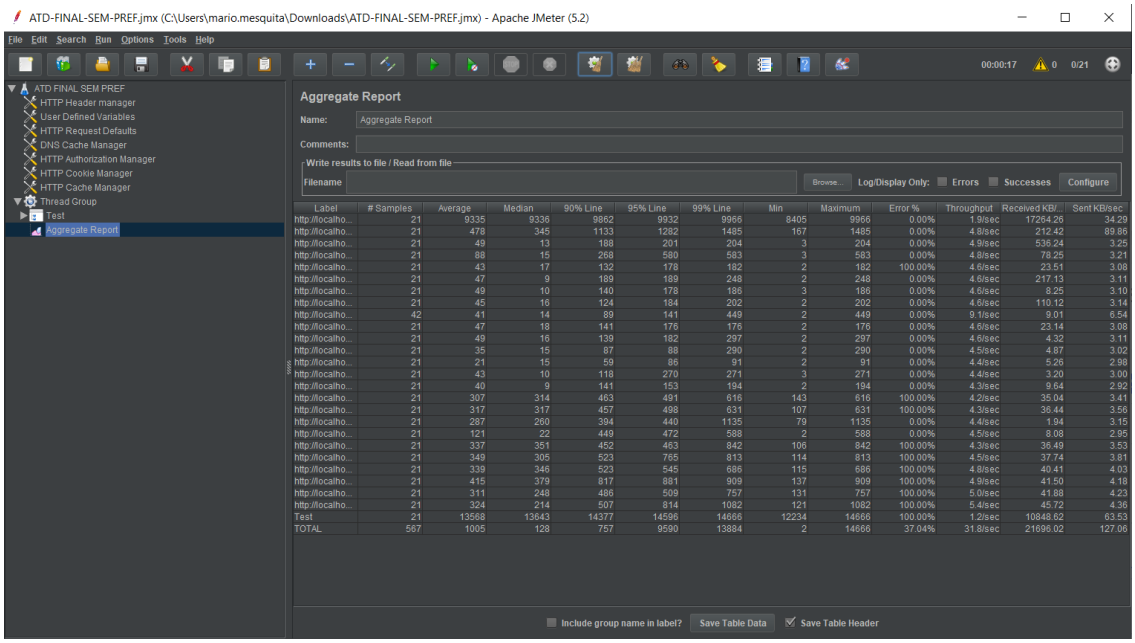


Figura 55 - Resultado inicial do Jmeter

Como se pode observar na figura 55, o tempo médio de resposta que cada utilizador obteve foi superior ao apresentado pelo Taurus, fixando-se nos 13,5 segundos. No entanto, os valores obtidos na avaliação no fim do projeto já se mostraram muito idênticos entre as duas ferramentas de teste.

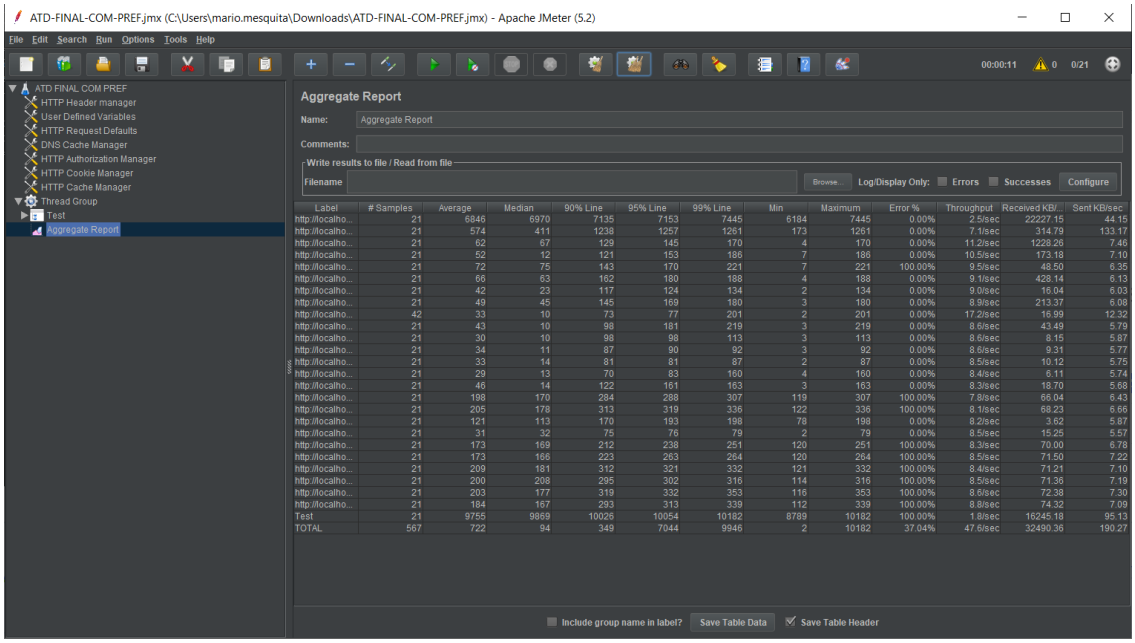


Figura 56 - Resultado final do Jmeter

Na figura 56 pode-se observar o valor médio obtido pelo teste que ronda os 9,7 segundos, tornando assim possível comprovar, mais uma vez, o sucesso das intervenções na aplicação do Atendimento.

6 Conclusões e trabalho futuro

O desenvolvimento deste projeto permitiu o estudo e análise de um problema muitas vezes enfrentado na sociedade. A otimização da performance Web é, e vai continuar a ser, uma grande mais valia, podendo determinar a escolha entre duas aplicações. No entanto, a performance não é uma ciência exata e, tal como se verificou no decorrer deste projeto, está em constante evolução. Umas das tecnologias utilizadas na avaliação de desempenho da aplicação sofreu uma atualização, cuja documentação mostra algumas mudanças significativas. Considerou-se esta atualização como uma medida positiva, uma vez que significa que as empresas, nomeadamente a Google, mostram cada vez mais interesse no tópico deste trabalho e cada vez mais tentam melhorar as suas ferramentas. Ao melhorar os recursos que promovem e ao facilitar a análise de desempenho Web, está-se a atrair mais opiniões e a consciencializar o mundo do quão importante é este assunto.

No início do desenvolvimento deste trabalho, foi decidido, em conjunto com responsáveis da empresa, que se iniciasse uma pesquisa de *software* capaz de avaliar o desempenho das aplicações e que fornecesse informação esclarecedora a partir dessa avaliação. Esta procura nem sempre foi fácil, uma vez que a maior parte do *software* encontrado era pago e não havia condições para testar o potencial valor de cada um deles. No entanto, alguns programas promoviam o seu trabalho com ofertas de subscrição limitadas, pelo que foi possível testar e avaliar com sucesso um grande número de tecnologias relevantes que, por consequência, foram relatadas aos responsáveis da empresa interessados na pesquisa. Algumas destas tecnologias, no final do projeto, mostraram-se importantes na avaliação e na recolha de dados relativos à performance da aplicação estudada.

De seguida, estudaram-se os casos de possíveis problemas que surgem quando se fazem análises de desempenho a aplicações e quais as metodologias a abordar para facilitar a sua resolução.

Uma vez concluído o estudo de tecnologias relevantes e de metodologias a aplicar na resolução de problemas, obteve-se acesso ao código da aplicação do Atendimento, que carecia de cuidados a nível de desempenho, tanto no seu *frontend* como no seu *backend.* Como o trabalho incide na performance Web, o foco foi a otimização das suas páginas e gestão dos seus recursos, de forma a obter uma melhor resposta dos seus conteúdos. Inicialmente, submeteu-se a aplicação a testes das respetivas tecnologias para promover a descoberta de problemas. Uma vez encontrados, e como já havia conhecimento de metodologias a aplicar, os problemas foram sendo resolvidos e atenuados. O resultado final da aplicação com as diversas metodologias aplicadas apresentava sinais positivos de melhoria que foram confirmados com uma nova sessão de testes por parte das tecnologias anteriormente definidas.

Para o autor, o desenvolvimento deste projeto revelou-se muito importante para o crescimento pessoal e profissional do mesmo. Espera-se que com o conhecimento aqui adquirido seja

possível transformar mais aplicações e obter mais resultados positivos. É da intenção do autor continuar a investigar sobre este assunto e procurar sempre novas formas de enfrentar os impasses e solucioná-los de forma exemplar.

Relativamente às limitações encontradas no desenvolvimento do projeto, salienta-se a impossibilidade de poder recorrer a uma ferramenta de testes que permitisse a criação de testes em maior escala para submeter a aplicação a uma carga ainda maior e obter mais resultados. Lamenta-se também o facto de não ser possível questionar os elementos da equipa de desenvolvimento do Atendimento acerca do trabalho aqui desenvolvido, uma vez que o projeto foi terminado em regime de teletrabalho, devido à pandemia que nos afeta. À data da escrita deste documento, ainda não foi possível juntar as versões da aplicação. O regime de teletrabalho obrigou também a recorrer à VPN da empresa para aceder à rede privada, o que por vezes dificultou a avaliação do desempenho da aplicação, uma vez que a velocidade da rede era mais limitada.

O trabalho futuro, poderá passar por certas intervenções que não foram aplicadas (como a minificação), que mereciam ser reavaliadas, uma vez que aplicadas no ambiente de produção poderiam trazer resultados benéficos no desempenho final da aplicação. Por fim, conclui-se que os resultados do Atendimento mostraram melhorias e as metodologias aqui descritas poderiam ser aplicadas a outras aplicações da Glintt.

Referências

[1] "15 Web Performance Stats You Should Know." [Online]. Available: https://cdnify.com/blog/15-web-performance-stats/. [Accessed: 11-Feb-2020].

[2] "Why Performance Matters | Web Fundamentals | Google Developers." [Online]. Available: https://developers.google.com/web/fundamentals/performance/why-performance-matters. [Accessed: 05-Nov-2019].

[3] "Why Performance Matters | Web Fundamentals | Google Developers." [Online]. Available: https://developers.google.com/web/fundamentals/performance/why-performance-matters#performance_is_about_retaining_users. [Accessed: 13-Jan-2020].

[4] "Driving user growth with performance improvements - Pinterest Engineering Blog - Medium." [Online]. Available: https://medium.com/pinterest-engineering/driving-user-growth-with-performance-improvements-cfc50dafadd7. [Accessed: 13-Jan-2020].

[5] "How the BBC builds websites that scale | Creative Bloq." [Online]. Available: https://www.creativebloq.com/features/how-the-bbc-builds-websites-that-scale. [Accessed: 13-Jan-2020].

[6] "How One Second Could Cost Amazon $1.6 Billion In Sales." [Online]. Available: https://www.fastcompany.com/1825005/how-one-second-could-cost-amazon-16-billion-sales. [Accessed: 14-Jan-2020].

[7] "Why Google is the Best Search Engine & Why You Should Care | Tower." [Online]. Available: https://www.towermarketing.net/blog/google-best-search-engine/. [Accessed: 14-Jan-2020].

[8] "Website Performance Impacts on your businness." [Online]. Available: https://www.dareboost.com/en/webperf-impacts. [Accessed: 14-Jan-2020].

[9] P. A. Koen, H. M. J. Bertels, and E. J. Kleinschmidt, "Managing the Front End of Innovation—Part II," doi: 10.5437/08956308X5703199.

[10] A. Brem and K. I. Voigt, "Integration of market pull and technology push in the corporate front end and innovation management-Insights from the German software industry," *Technovation*, vol. 29, no. 5, pp. 351–367, May 2009, doi: 10.1016/j.technovation.2008.06.003.

[11] "Mobile stress: Slower web pages lead to increased user frustration and lower engagement | Radware Blog." [Online]. Available: https://blog.radware.com/applicationdelivery/wpo/2013/12/slower-web-pages-user-frustration/. [Accessed: 16-Jan-2020].

[12] "Websites That Suck Increase Stress - Neuromarketing." [Online]. Available:

https://www.neurosciencemarketing.com/blog/articles/web-stress.htm. [Accessed: 16-Jan-2020].

[13] “Is Your Mobile Website Stressing People Out? A Simple Guide to Improving Page Load Time.” [Online]. Available: https://blog.hubspot.com/marketing/mobile-website-load-faster. [Accessed: 16-Jan-2020].

[14] “Basic Performance Tips.” [Online]. Available: https://developer.apple.com/library/archive/documentation/Performance/Conceptual/PerformanceOverview/BasicTips/BasicTips.html#//apple_ref/doc/uid/TP40001410-CH204-BCIFDFAA. [Accessed: 20-Jan-2020].

[15] “An Introduction to Perceived Performance | Treehouse Blog.” [Online]. Available: https://blog.teamtreehouse.com/perceived-performance. [Accessed: 20-Jan-2020].

[16] “Why Perceived Performance Matters, Part 1: The Perception Of Time — Smashing Magazine.” [Online]. Available: https://www.smashingmagazine.com/2015/09/why-performance-matters-the-perception-of-time/#the-need-for-performance-optimization-the-20-rule. [Accessed: 20-Jan-2020].

[17] “Literature Review on Reaction Time.” [Online]. Available: http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/biae.clemson.edu/bpc/bp/Lab/110/reaction.htm#Mean Times. [Accessed: 07-Jan-2020].

[18] “The Illusion of Speed - Paul Bakaus’ blog.” [Online]. Available: https://paulbakaus.com/tutorials/performance/the-illusion-of-speed/. [Accessed: 23-Jan-2020].

[19] “Why Performance Matters, Part 2: Perception Management — Smashing Magazine.” [Online]. Available: https://www.smashingmagazine.com/2015/11/why-performance-matters-part-2-perception-management/#preemptive-start. [Accessed: 06-Nov-2019].

[20] “Why Performance Matters, Part 2: Perception Management — Smashing Magazine.” [Online]. Available: https://www.smashingmagazine.com/2015/11/why-performance-matters-part-2-perception-management/#early-completion. [Accessed: 06-Nov-2019].

[21] “GTmetrix | Website Speed and Performance Optimization.” [Online]. Available: https://gtmetrix.com/. [Accessed: 23-Feb-2020].

[22] “PageSpeed Insights.” [Online]. Available: https://developers.google.com/speed/pagespeed/insights/?hl=pt-PT. [Accessed: 23-Feb-2020].

[23] “Prefetching/Preloading and prerendering content with HTML.” [Online]. Available: https://jack.ofspades.com/prefetching-preloading-and-prerendering-content-with-html/. [Accessed: 23-Jan-2020].

[24] “Preload, Prefetch And Priorities in Chrome - reloading - Medium.” [Online]. Available: https://medium.com/reloading/preload-prefetch-and-priorities-in-chrome-776165961bbf. [Accessed: 23-Jun-2020].

[25] “Tree Shaking | webpack.” [Online]. Available: https://webpack.js.org/guides/tree-

shaking/. [Accessed: 23-Feb-2020].

[26] “Lazy Loading Images and Video | Web Fundamentals | Google Developers.” [Online]. Available: https://developers.google.com/web/fundamentals/performance/lazy-loading-guidance/images-and-video. [Accessed: 23-Feb-2020].

[27] “What is JMeter? Introduction & Uses.” [Online]. Available: https://www.guru99.com/introduction-to-jmeter.html. [Accessed: 11-Nov-2019].

[28] “About LoadNinja | LoadNinja Documentation.” [Online]. Available: https://support.smartbear.com/loadninja/docs/general-info/about.html. [Accessed: 11-Nov-2019].

[29] “Load Testing Tool | NeoLoad by Neotys.” [Online]. Available: https://www.neotys.com/neoload/overview. [Accessed: 23-Feb-2020].

[30] “BlazeMeter, an Enterprise-grade Continuous Testing Platform.” [Online]. Available: https://www.blazemeter.com/. [Accessed: 23-Feb-2020].

[31] “Cloud Based Load & Performance Testing | Flood.io.” [Online]. Available: https://flood.io/. [Accessed: 23-Feb-2020].

[32] “Gatling Open-Source Load Testing Documentation – For DevOps and CI/CD.” [Online]. Available: https://gatling.io/docs/current/. [Accessed: 23-Feb-2020].

[33] “Load Impact | Performance testing for DevOps teams.” [Online]. Available: https://loadimpact.com/. [Accessed: 23-Feb-2020].

[34] “Load Testing Your API with Postman.” [Online]. Available: https://blog.loadimpact.com/load-testing-with-postman-collections. [Accessed: 23-Feb-2020].

[35] “Load Testing & Website Performance Tools - LoadView.” [Online]. Available: https://www.loadview-testing.com/. [Accessed: 23-Feb-2020].

[36] “SaaS and On-Premise Load Testing - OctoPerf.” [Online]. Available: https://octoperf.com/. [Accessed: 23-Feb-2020].

[37] “Taurus: A New Star in the Test Automation Tools Constellation | BlazeMeter.” [Online]. Available: https://www.blazemeter.com/blog/taurus-new-star-test-automation-tools-constellation/. [Accessed: 23-Feb-2020].

[38] “Features | SmartMeter.” [Online]. Available: https://www.smartmeter.io/features. [Accessed: 23-Feb-2020].

[39] “LoadRunner Alternative - WebLOAD.” [Online]. Available: https://www.radview.com/loadrunner-replacement/. [Accessed: 23-Feb-2020].

[40] “Lighthouse | Tools for Web Developers | Google Developers.” [Online]. Available: https://developers.google.com/web/tools/lighthouse. [Accessed: 23-Jun-2020].

[41] “How to Speed Up Your Web App and Improve Website Performance.” [Online]. Available: https://stxnext.com/blog/2019/05/16/web-app-speed-website-

performance/. [Accessed: 28-Jan-2020].

[42] “SQL vs NoSQL: What’s the difference?” [Online]. Available: https://www.guru99.com/sql-vs-nosql.html. [Accessed: 10-Feb-2020].

[43] “What does ACID mean in Database Systems? | Database.Guide.” [Online]. Available: https://database.guide/what-is-acid-in-databases/. [Accessed: 23-Feb-2020].

[44] “SQL vs NoSQL: High-Level Differences – Big Data.” [Online]. Available: http://bigdata.black/infrastructure/storage/sql-nosql-differences/. [Accessed: 10-Feb-2020].

[45] “First Contentful Paint | Tools for Web Developers | Google Developers.” [Online]. Available: https://developers.google.com/web/tools/lighthouse/audits/first-contentful-paint. [Accessed: 23-Feb-2020].

[46] “First Meaningful Paint | Tools for Web Developers | Google Developers.” [Online]. Available: https://developers.google.com/web/tools/lighthouse/audits/first-meaningful-paint. [Accessed: 23-Feb-2020].

[47] “What is page load time and why is it important? | BigCommerce.” [Online]. Available: https://www.bigcommerce.com/ecommerce-answers/what-page-load-time-and-why-it-important/. [Accessed: 23-Feb-2020].

[48] “Need For Speed: Top 10 Web Performance Metrics You Must Monitor.” [Online]. Available: https://www.namogoo.com/blog/conversion-rate-optimization/top-10-web-performance-metrics/. [Accessed: 23-Feb-2020].

[49] “Metrics - WebPagetest Documentation.” [Online]. Available: https://sites.google.com/a/webpagetest.org/docs/using-webpagetest/metrics. [Accessed: 23-Feb-2020].

[50] “Taxa de rejeição - Analytics Ajuda.” [Online]. Available: https://support.google.com/analytics/answer/1009409?hl=pt-PT. [Accessed: 23-Feb-2020].

[51] “Time to Interactive.” [Online]. Available: https://web.dev/interactive/. [Accessed: 23-Feb-2020].

[52] “What are RPS (Requests per second)? - LoadImpact.” [Online]. Available: https://support.loadimpact.com/3.0/test-configuration/what-are-requests-per-second-rps/. [Accessed: 23-Feb-2020].

[53] “Understanding web page weight.” [Online]. Available: https://deviceatlas.com/blog/understanding-web-page-weight. [Accessed: 23-Feb-2020].

[54] “Definition of Error Rates | Chegg.com.” [Online]. Available: https://www.chegg.com/homework-help/definitions/error-rates-3. [Accessed: 23-Feb-2020].

[55] “What is Response Time Testing?” [Online]. Available: https://www.guru99.com/response-time-testing.html. [Accessed: 23-Feb-2020].

[56] “What is TCP connection time? | Performance metrics.” [Online]. Available: https://varvy.com/performance/tcp-connection-time.html. [Accessed: 23-Feb-2020].

[57] “Total Blocking Time (TBT).” [Online]. Available: https://web.dev/tbt/. [Accessed: 18-Nov-2019].

[58] “Improve Server Response Time | PageSpeed Insights | Google Developers.” [Online]. Available: https://developers.google.com/speed/docs/insights/Server#overview. [Accessed: 09-Jan-2020].

[59] “User-centric Performance Metrics | Web Fundamentals.” [Online]. Available: https://developers.google.com/web/fundamentals/performance/user-centric-performance-metrics#mapping_metrics_to_user_experience. [Accessed: 11-Nov-2019].

[60] “Reduce JavaScript Payloads with Tree Shaking | Web Fundamentals.” [Online]. Available: https://developers.google.com/web/fundamentals/performance/optimizing-javascript/tree-shaking. [Accessed: 18-Jun-2020].

[61] “The Cost Of JavaScript - Dev Channel - Medium.” [Online]. Available: https://medium.com/dev-channel/the-cost-of-javascript-84009f51e99e. [Accessed: 19-Jun-2020].

[62] “How To Optimize Your Site With GZIP Compression – BetterExplained.” [Online]. Available: https://betterexplained.com/articles/how-to-optimize-your-site-with-gzip-compression/. [Accessed: 19-Jun-2020].

[63] “What is a browser cache, and why is it important? | BigCommerce.” [Online]. Available: https://www.bigcommerce.com/ecommerce-answers/what-browser-cache-and-why-it-important/. [Accessed: 22-Jun-2020].

[64] “Single-page application vs. multiple-page application.” [Online]. Available: https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58. [Accessed: 23-Jun-2020].

[65] “Releases · GoogleChrome/lighthouse · GitHub.” [Online]. Available: https://github.com/GoogleChrome/lighthouse/releases. [Accessed: 23-Jun-2020].

[66] “Lighthouse performance scoring.” [Online]. Available: https://web.dev/performance-scoring/. [Accessed: 23-Jun-2020].

[67] “First Contentful Paint.” [Online]. Available: https://web.dev/first-contentful-paint/. [Accessed: 23-Jun-2020].

[68] “Speed Index.” [Online]. Available: https://web.dev/speed-index/. [Accessed: 23-Jun-2020].

[69] “First Meaningful Paint.” [Online]. Available: https://web.dev/first-meaningful-paint/. [Accessed: 23-Jun-2020].

[70] “Time to Interactive.” [Online]. Available: https://web.dev/interactive/. [Accessed: 23-Jun-2020].

[71] “First CPU Idle.” [Online]. Available: https://web.dev/first-cpu-idle/. [Accessed: 23-Jun-2020].

[72] “Reduce server response times (TTFB).” [Online]. Available: https://web.dev/time-to-first-byte/. [Accessed: 23-Jun-2020].

Anexos

Anexo A

No presente anexo serão apresentadas algumas imagens dos resultados dos testes de performance efetuadas ao site MaisFutebol.

Blazemeter

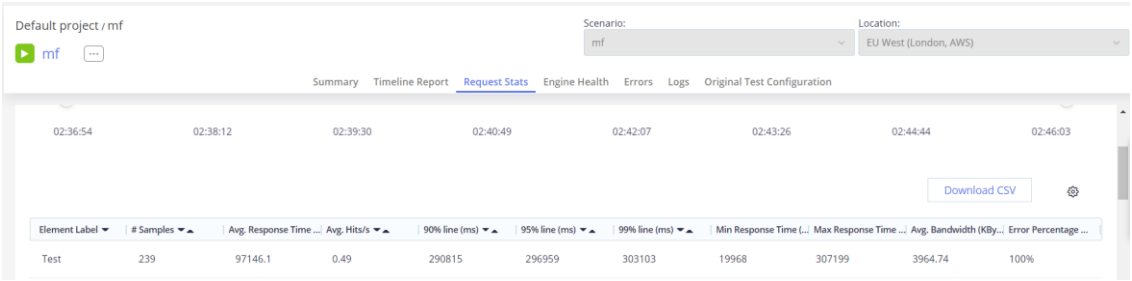


Figura 57 - Resultados da análise ao site MaisFutebol com 50 utilizadores

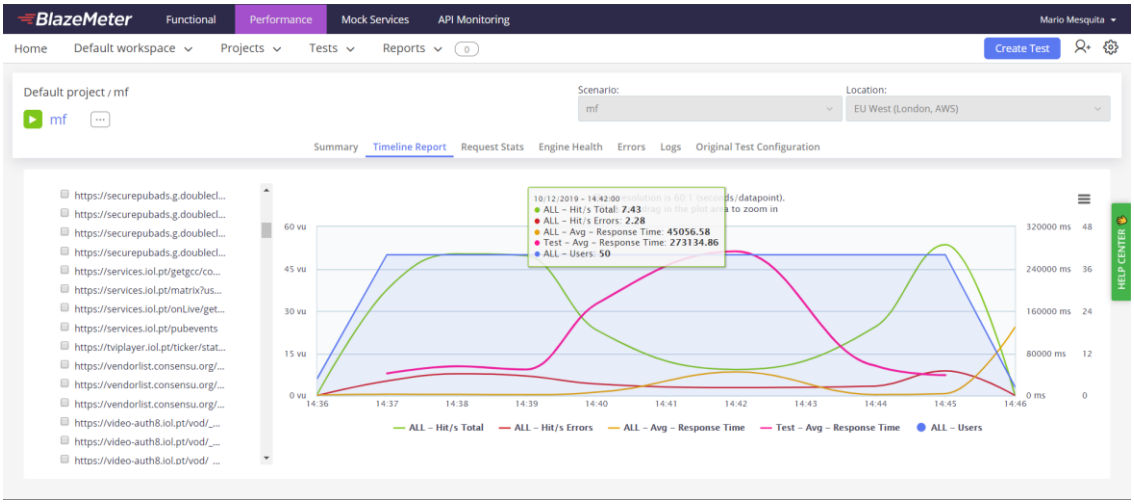


Figura 58 - Resultados da análise ao site MaisFutebol com 50 utilizadores numa linha temporal

Flood

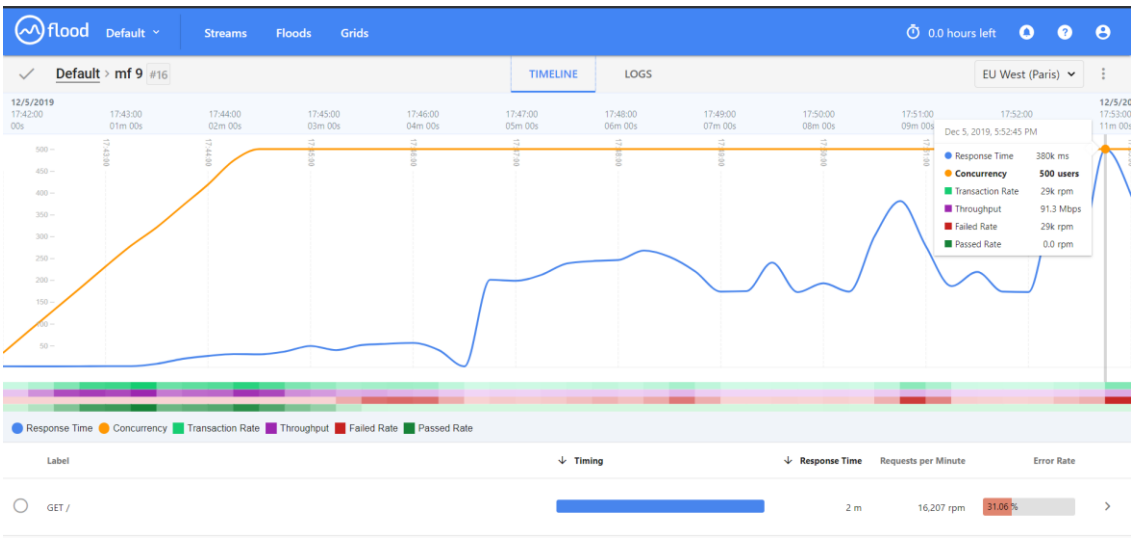


Figura 59 - Resultado da análise ao site MaisFutebol com 500 utilizadores

Gatling Frontline

Figura 60 - Resultado da análise ao site MaisFutebol com 200 utilizadores

JMeter

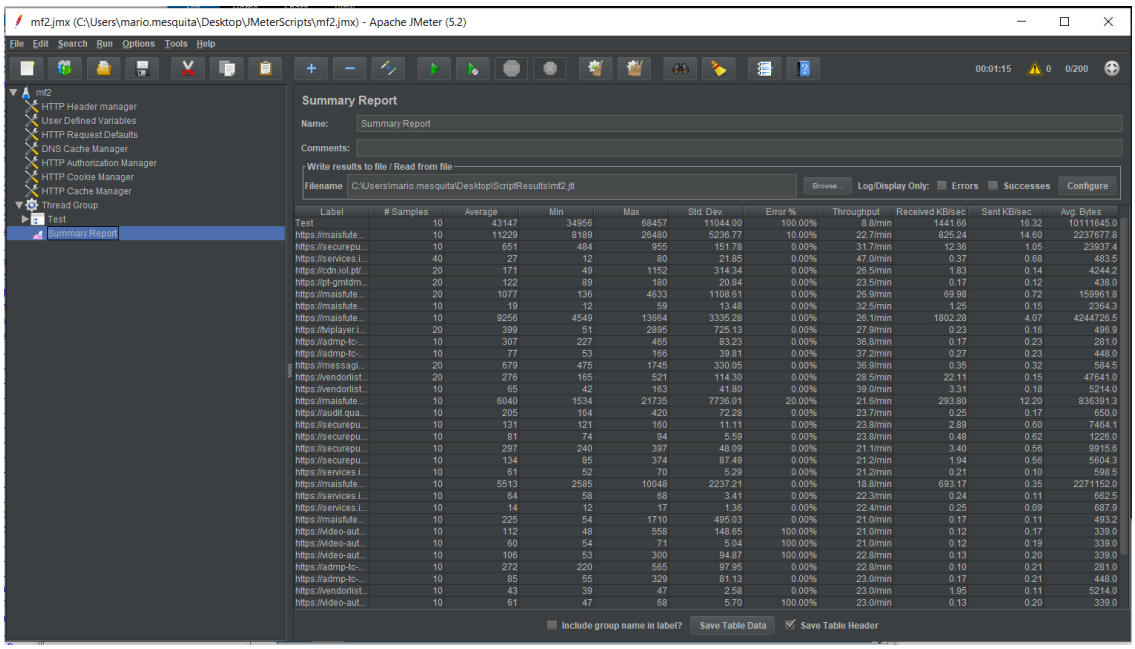


Figura 61 - Resultados da análise ao site MaisFutebol com 10 utilizadores

LoadImpact

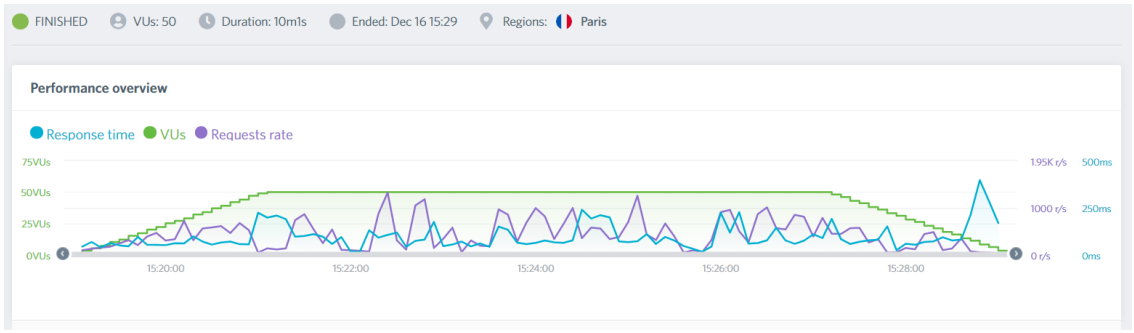


Figura 62 - Resultados da análise ao site MaisFutebol com 50 utilizadores

LoadView

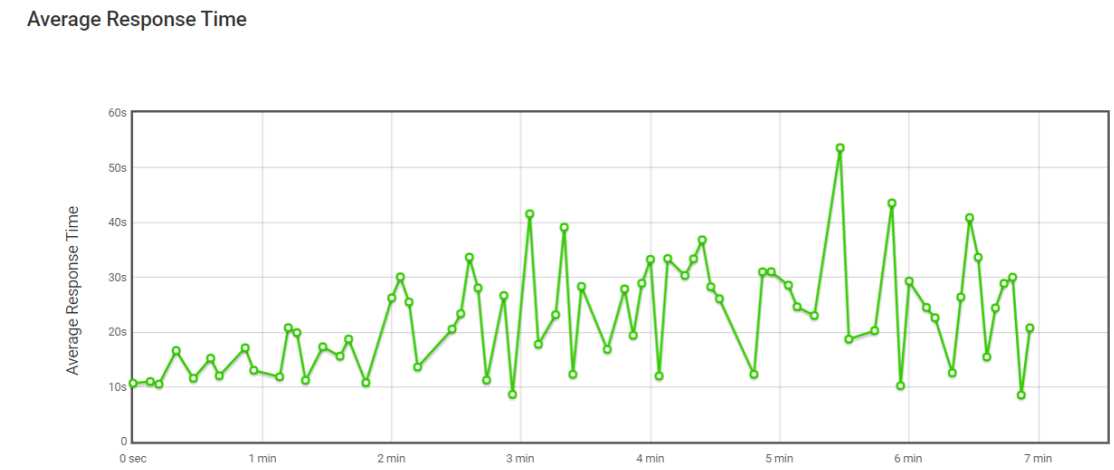


Figura 63 - Resultados da análise ao site MaisFutebol com 10 utilizadores

NeoLoad

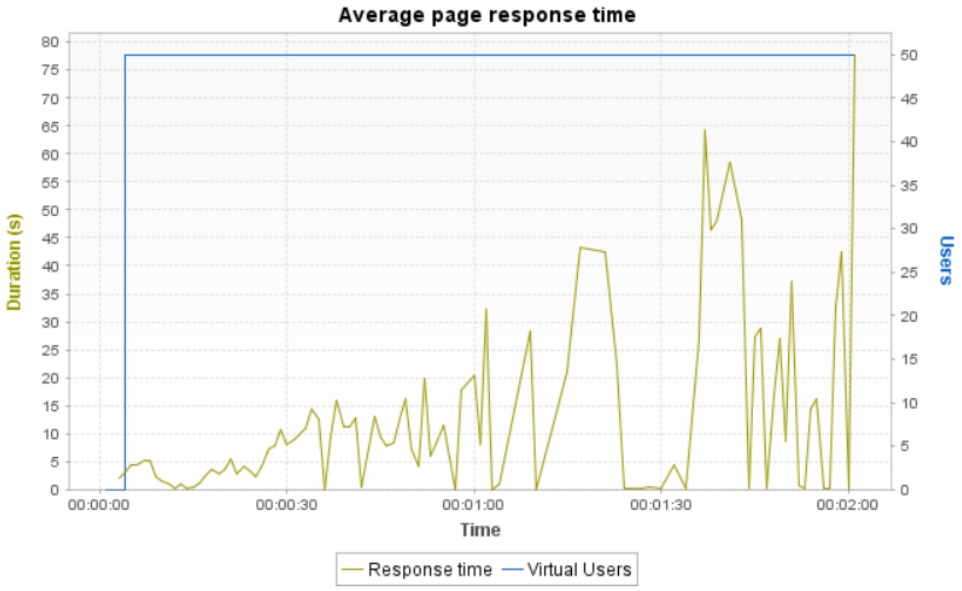


Figura 64 - Resultados da análise ao site MaisFutebol com 50 utilizadores

OctoPerf

Figura 65 - Resultados da análise ao site MaisFutebol com 50 utilizadores

Smartmeter.io

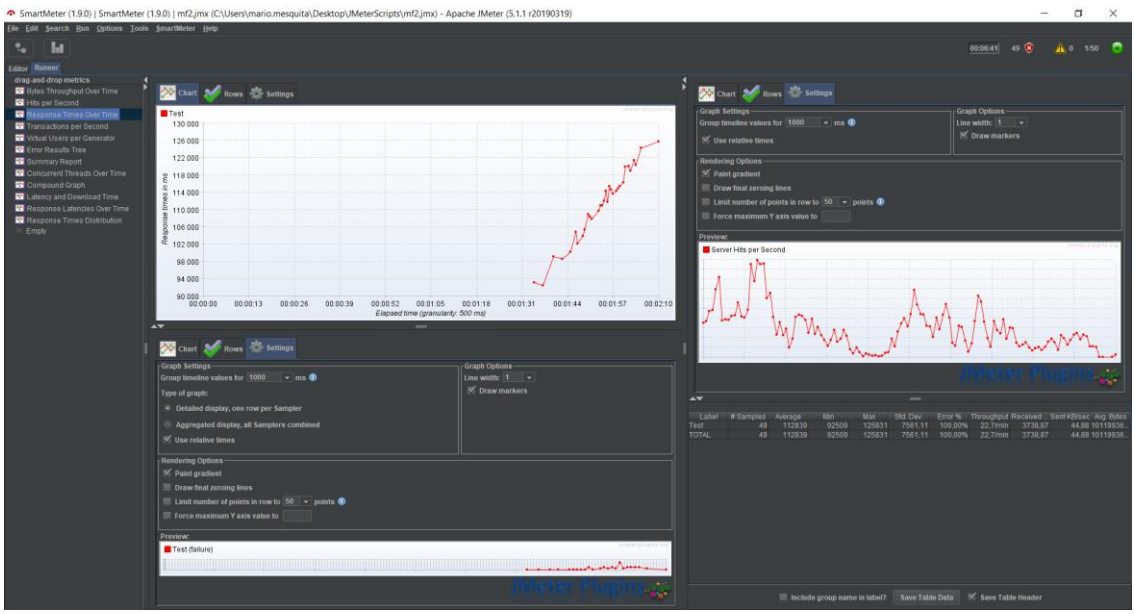


Figura 66 - Resultados da análise ao site MaisFutebol com 50 utilizadores

Taurus

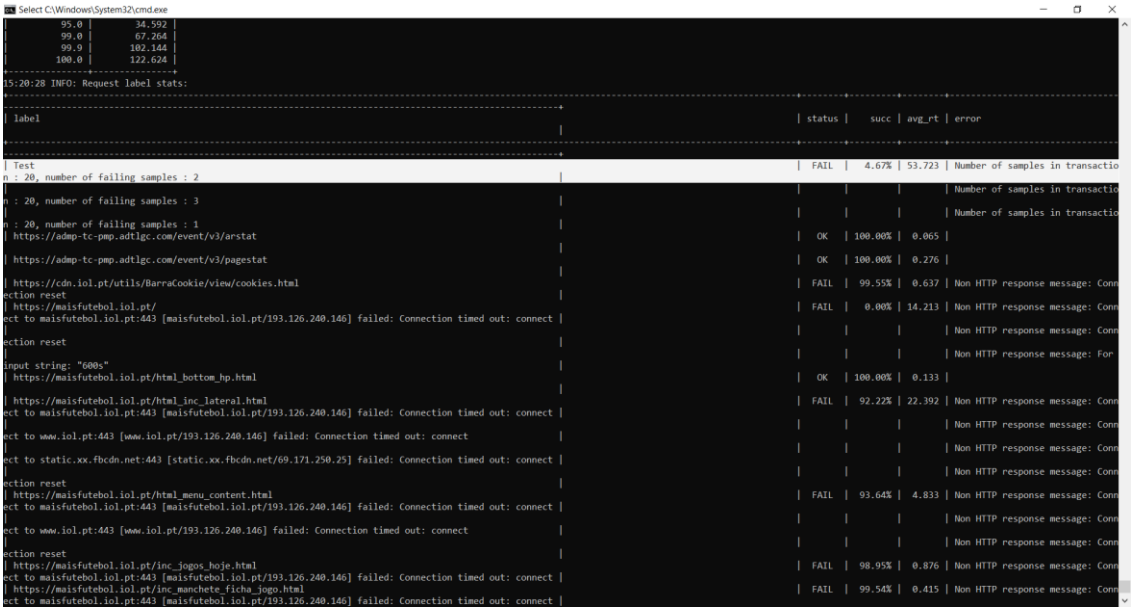


Figura 67 - Resultados da análise ao site MaisFutebol com 20 utilizadores

WebLOAD

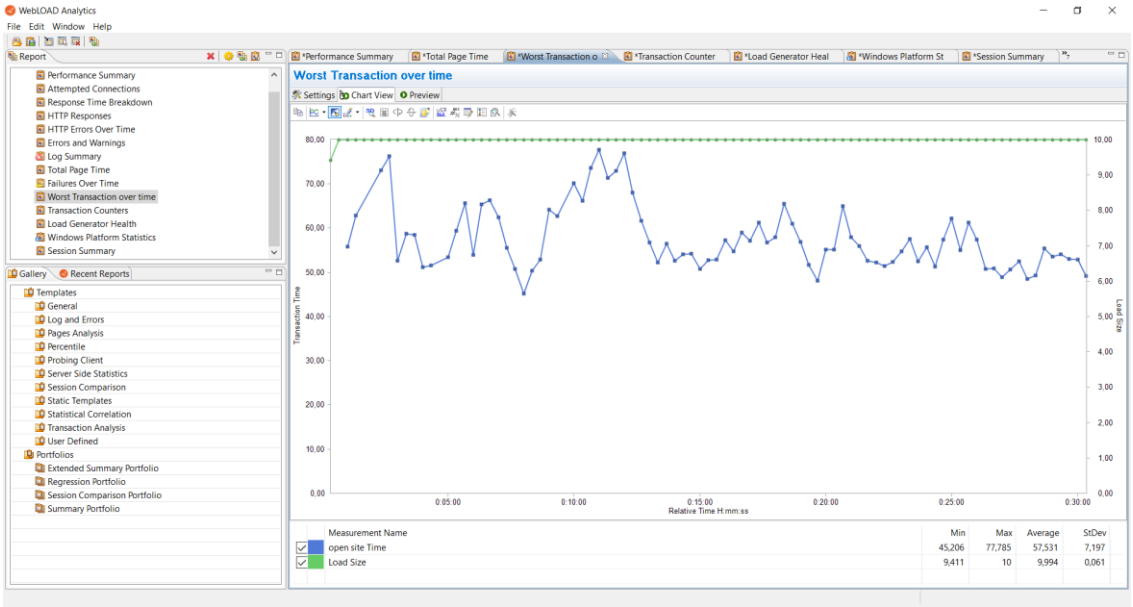


Figura 68 - Resultados da análise ao site MaisFutebol com 10 utilizadores

Anexo B

Imagens da aba *Network* do Google Chrome relativas ao Atendimento antes das intervenções

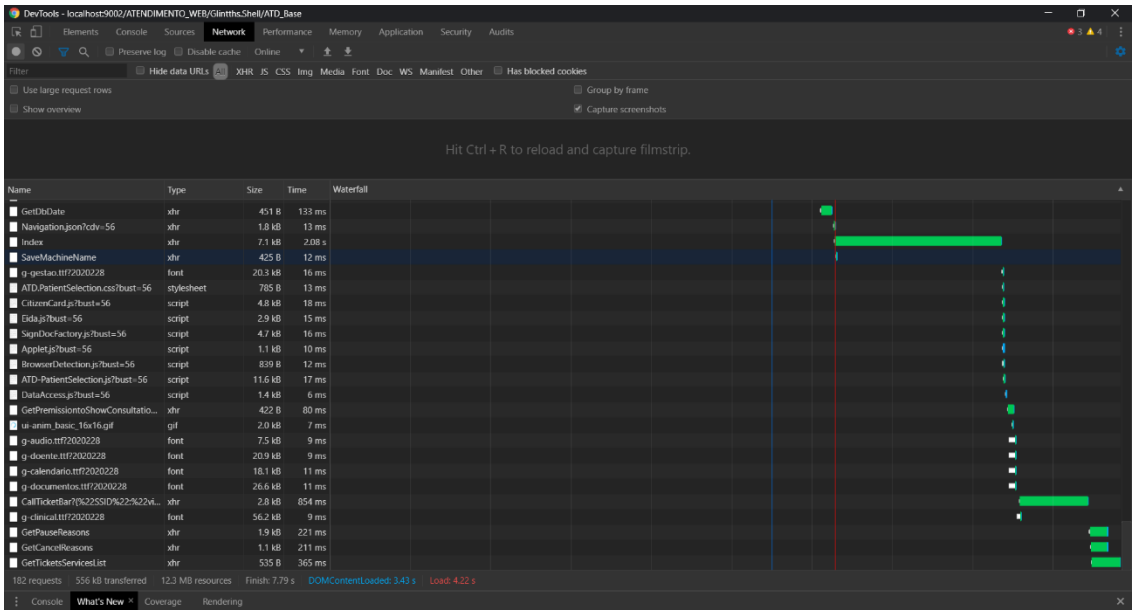


Figura 69 - Página inicial do Atendimento

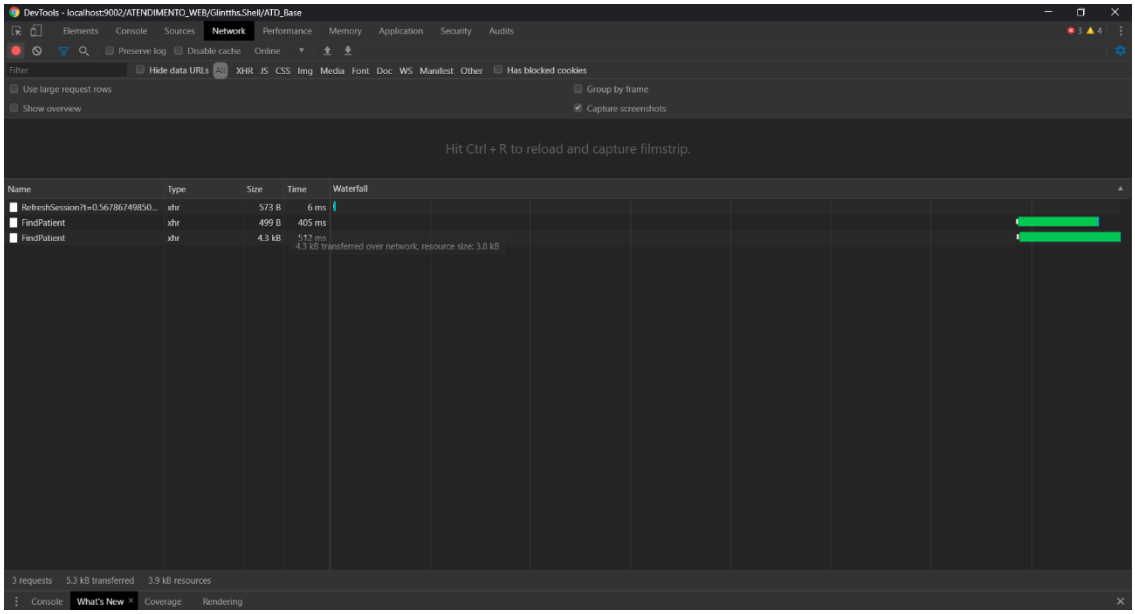


Figura 70 - Pesquisa de doentes

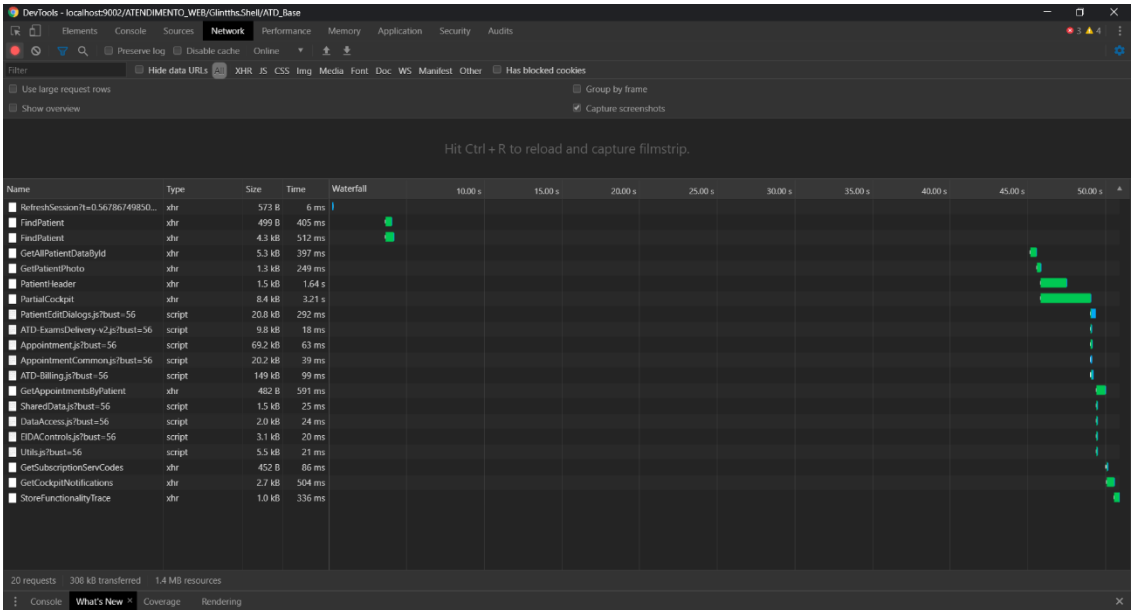


Figura 71 - Cockpit do doente

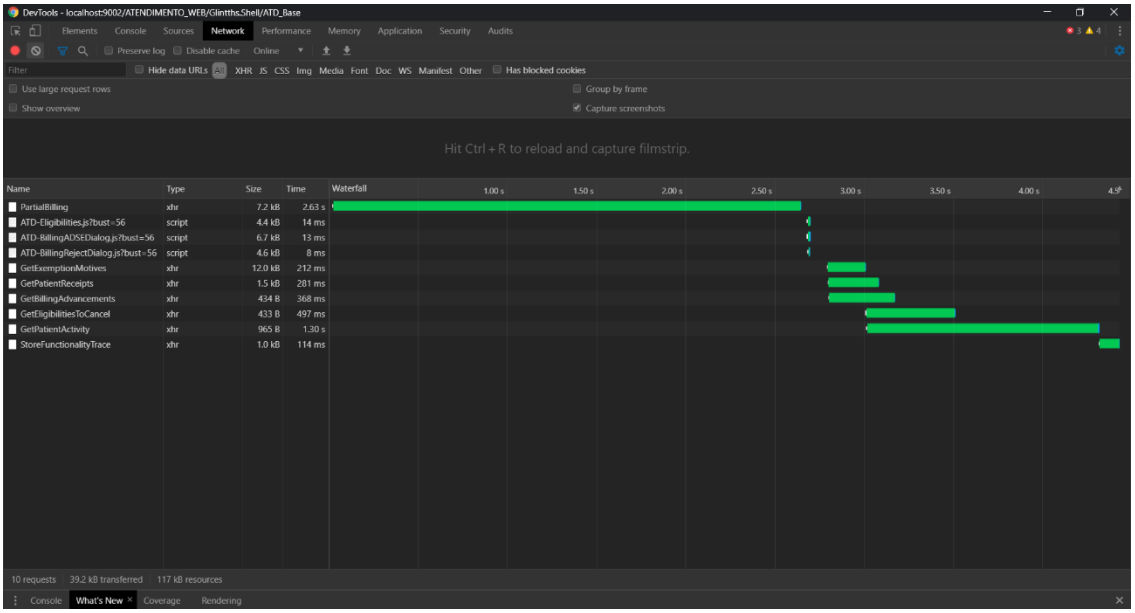


Figura 72 – Pagamento

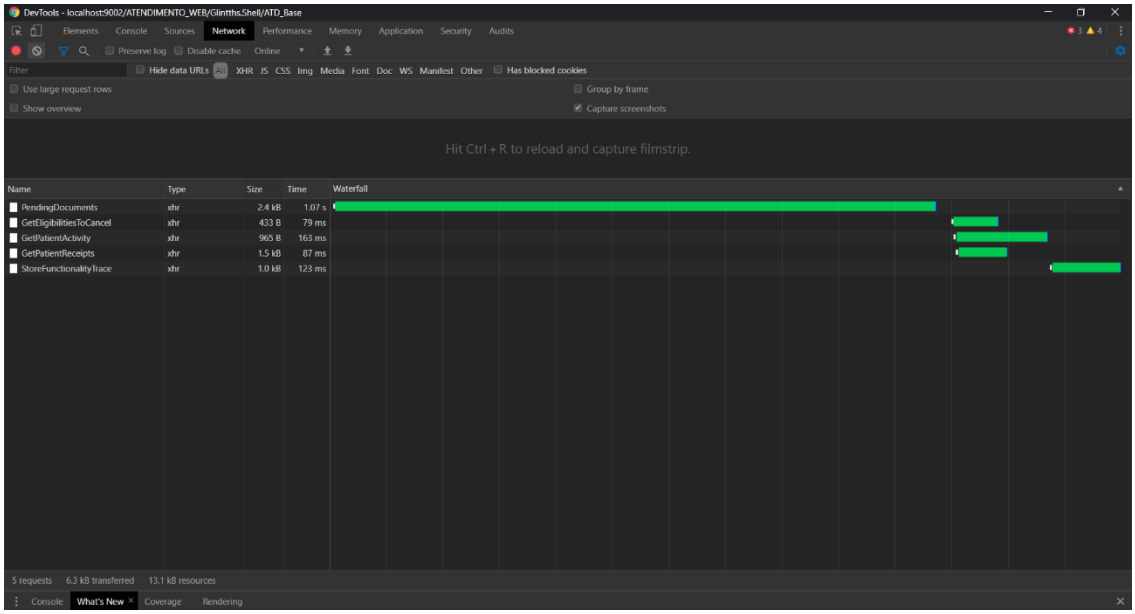


Figura 73- Pagamento de documentos pendentes

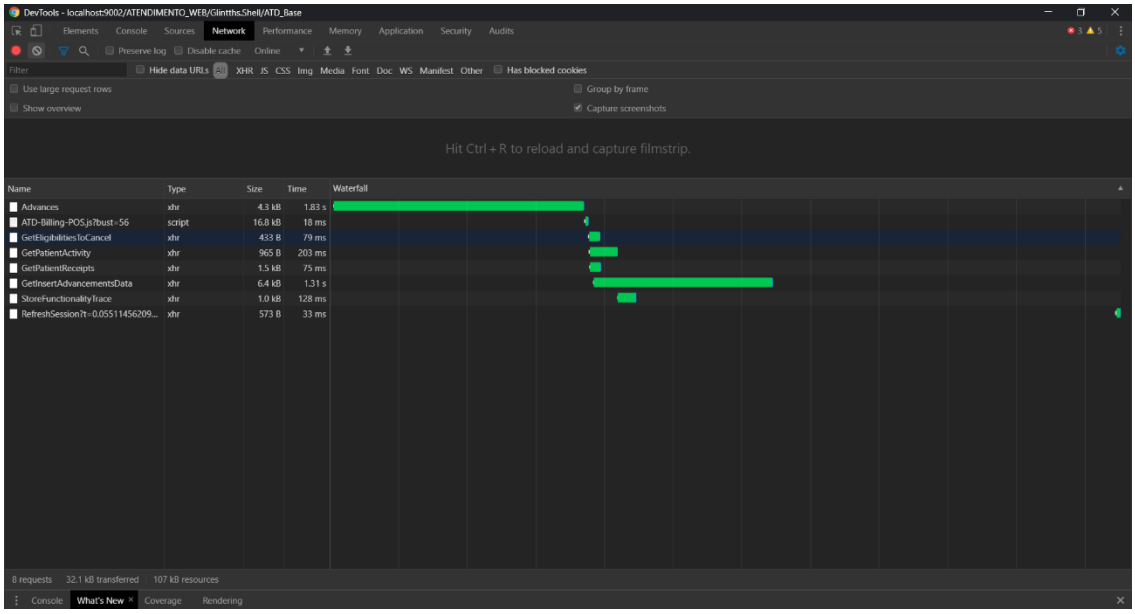


Figura 74 - Pagamentos de Cauções e adiantamentos

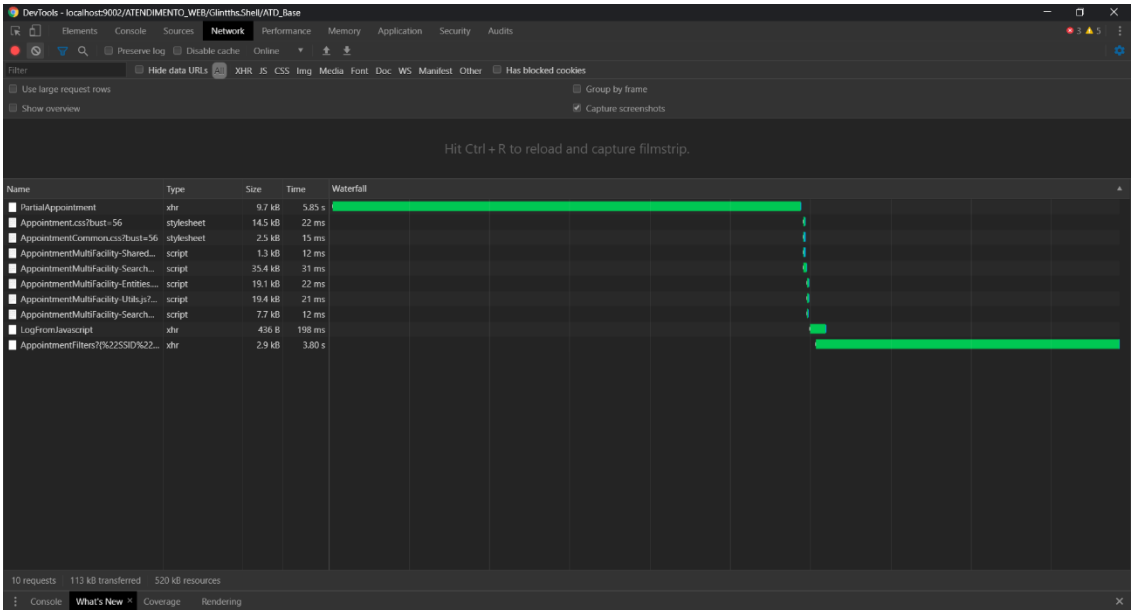


Figura 75 – Marcação

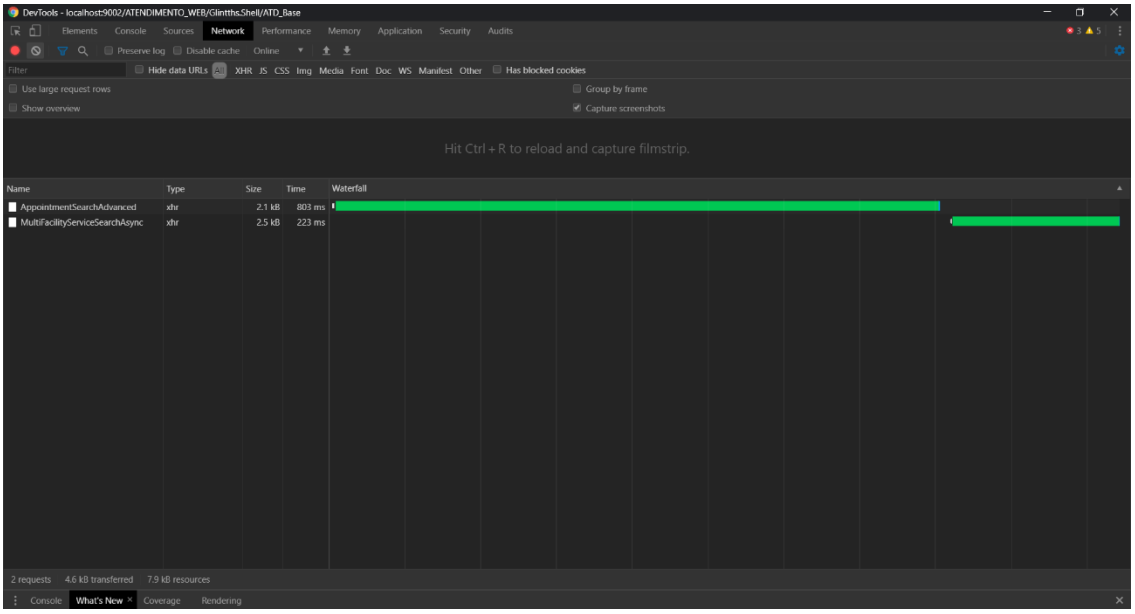


Figura 76 - Pesquisa avançada

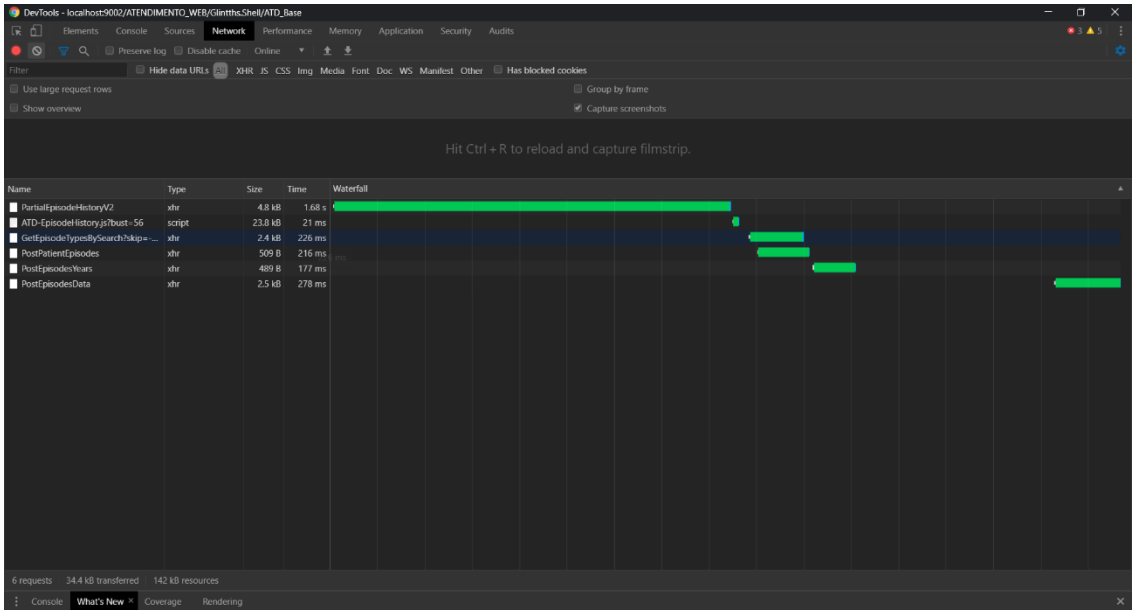


Figura 77 - Histórico

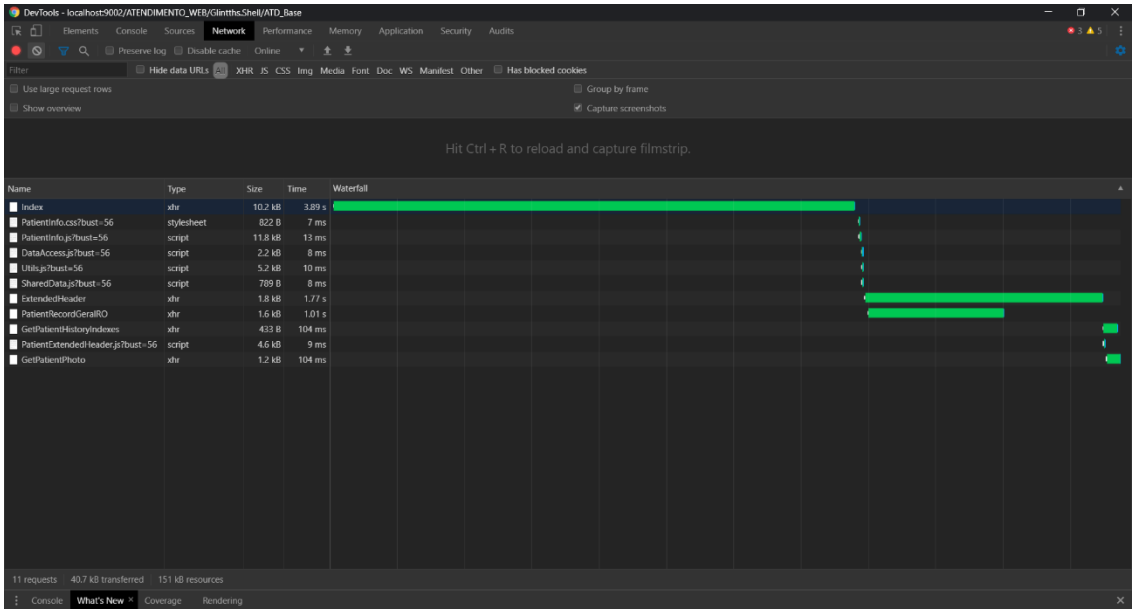


Figura 78 -Ficha de doente

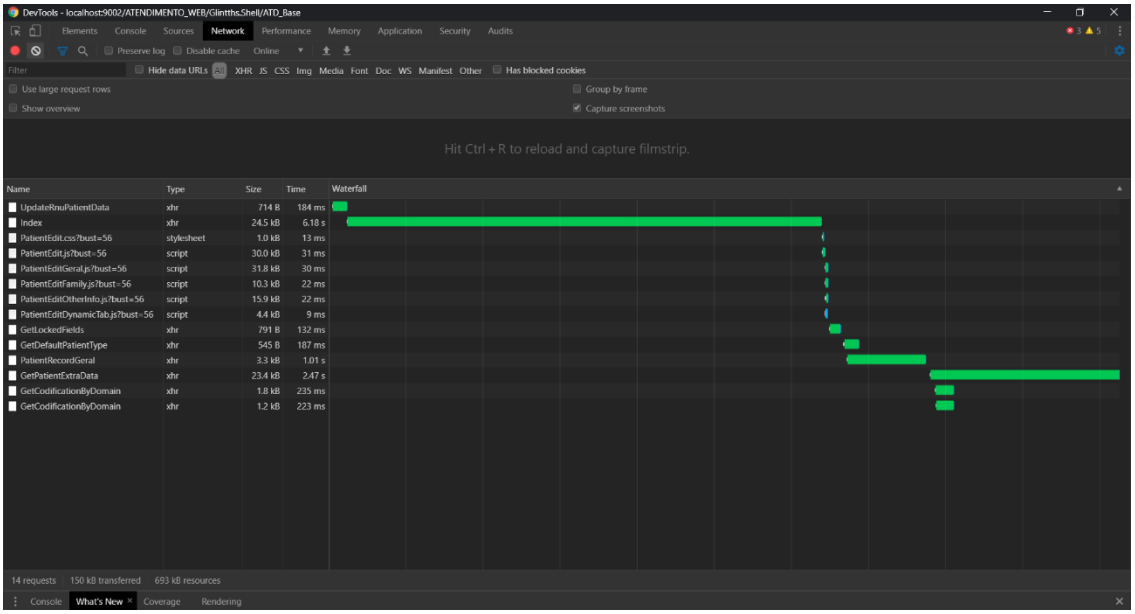


Figura 79 - Edição da ficha de doente

Imagens da aba *Network* do Google Chrome relativas ao Atendimento após as intervenções

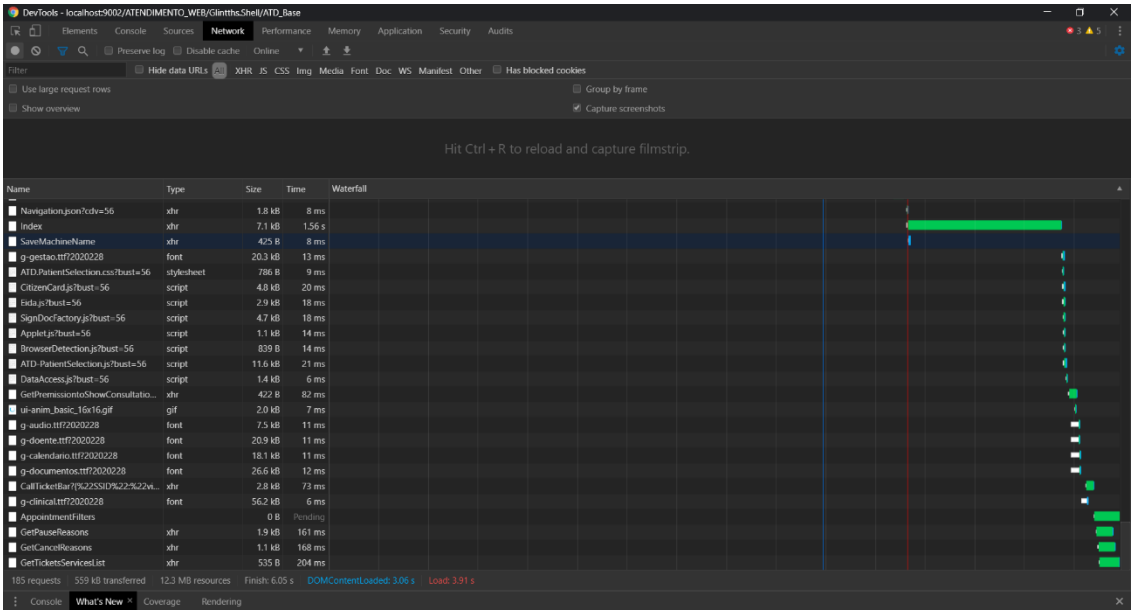


Figura 80 - Página inicial do Atendimento após intervenções

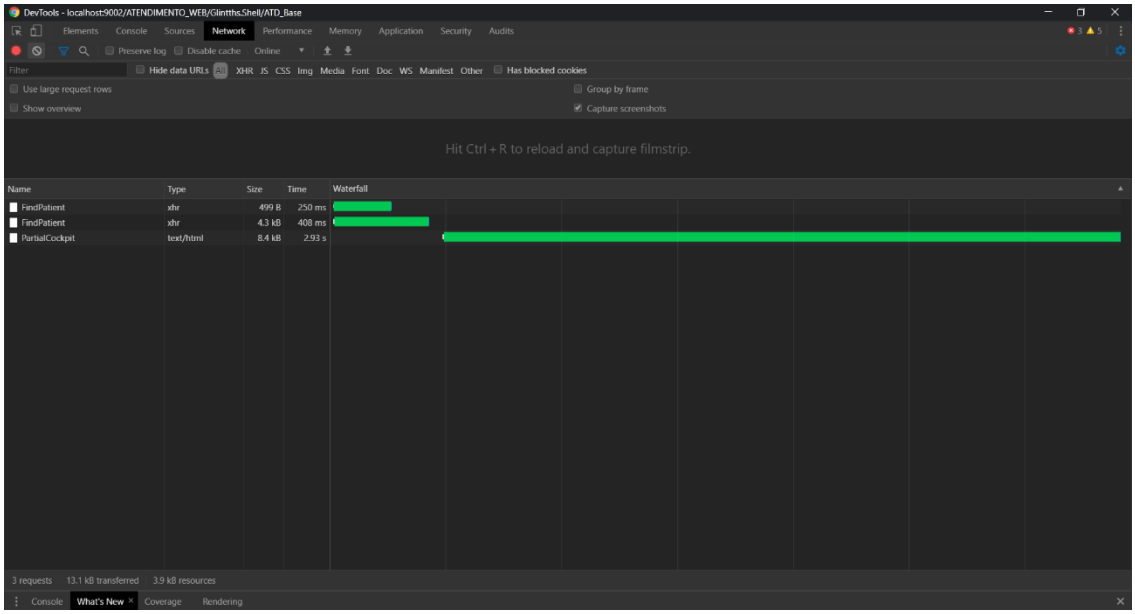


Figura 81 - Pesquisa de doentes após as intervenções

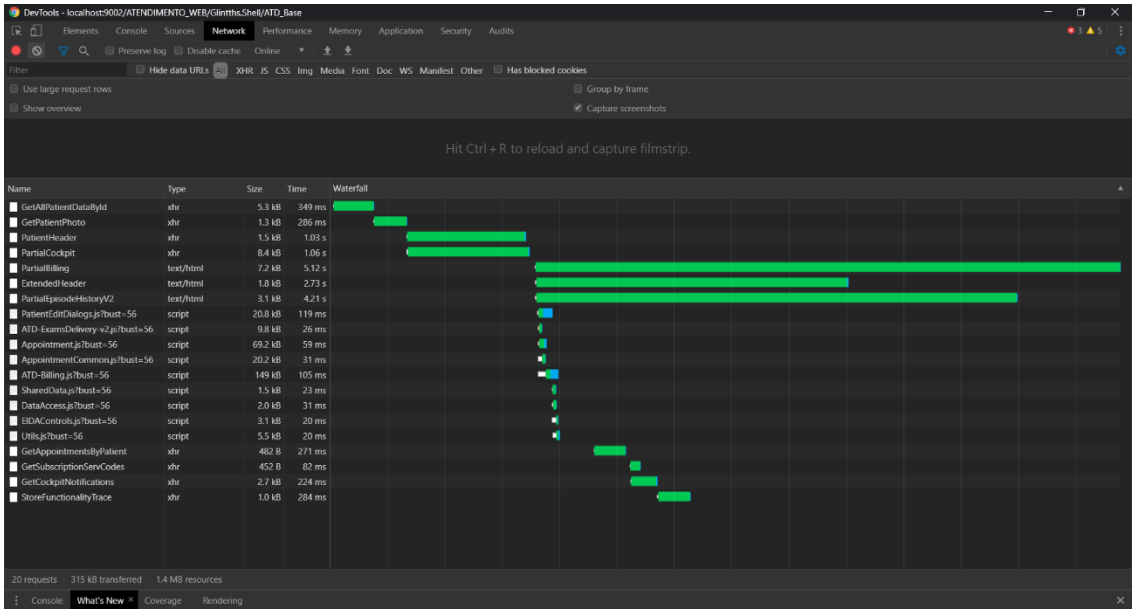


Figura 82 - Cockpit após intervenções

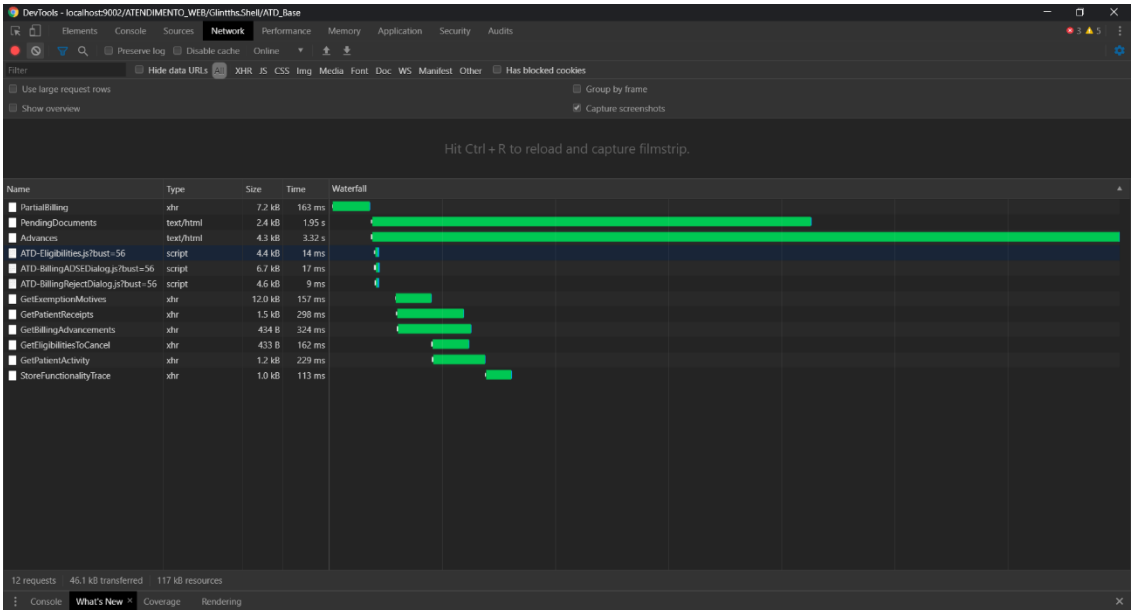


Figura 83 - Pagamento após intervenções

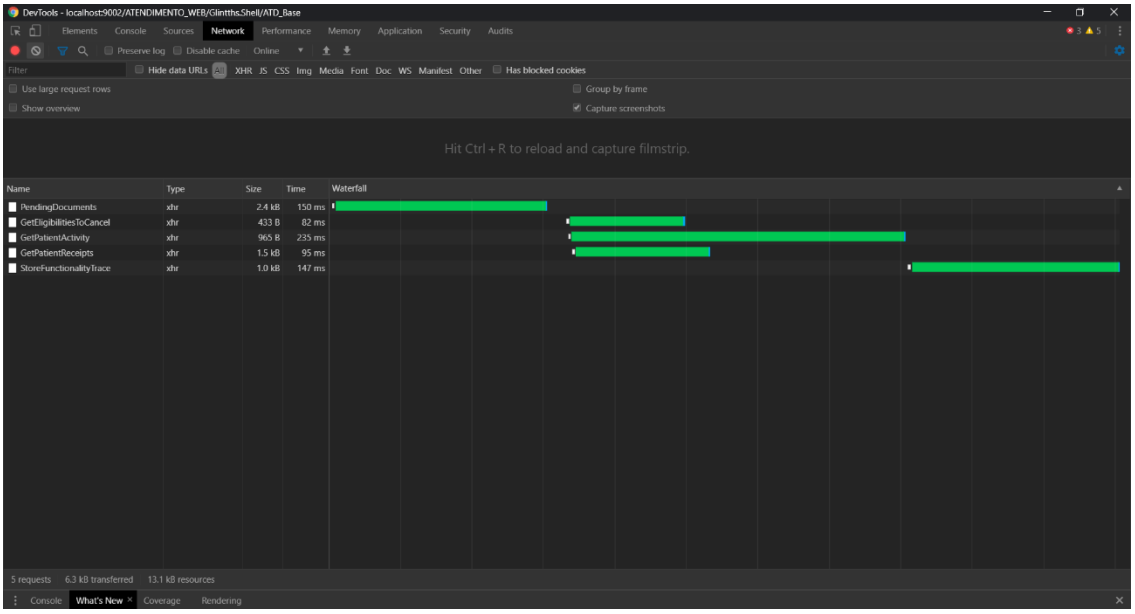


Figura 84 - Pagamento de documentos pendentes após intervenções

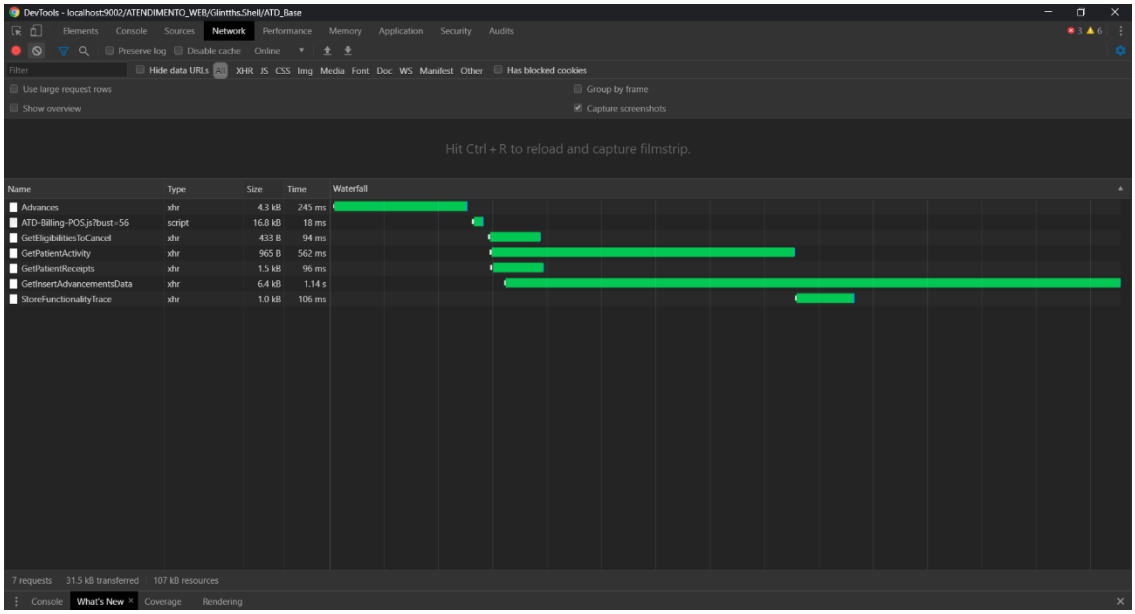


Figura 85 - Pagamentos de Cauções e adiantamentos após intervenções

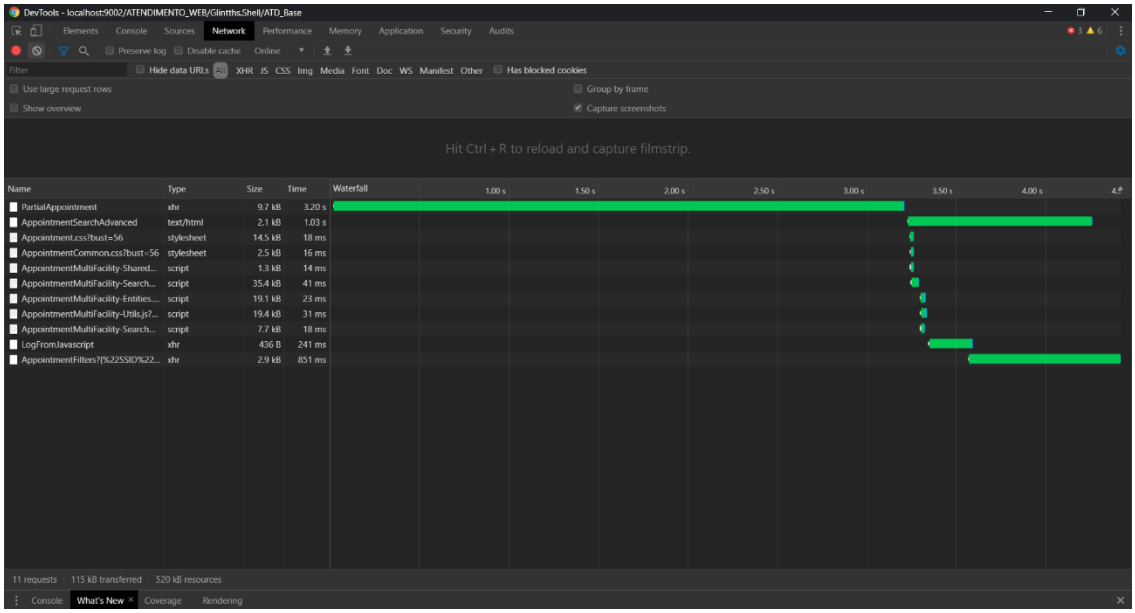


Figura 86 - Marcação após intervenções

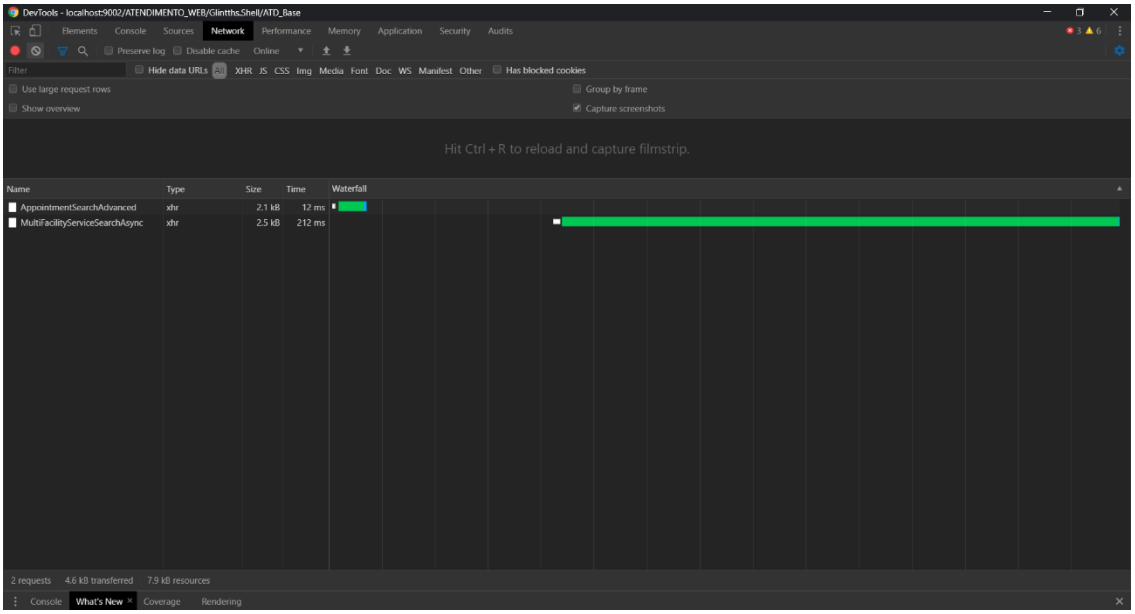


Figura 87 - Pesquisa avançada após intervenções

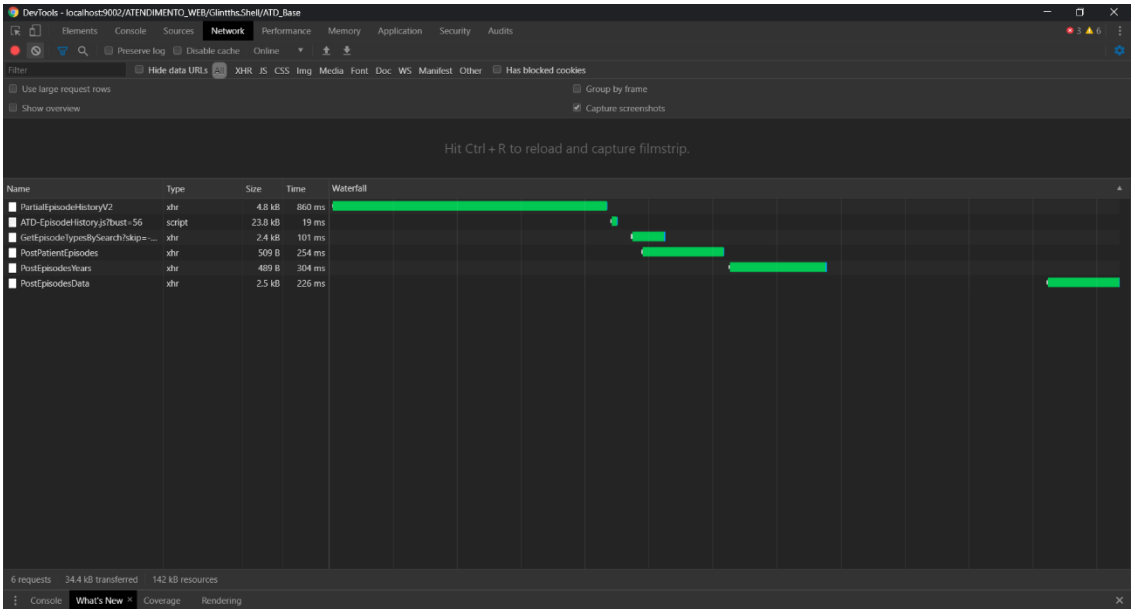


Figura 88 - Histórico após intervenções

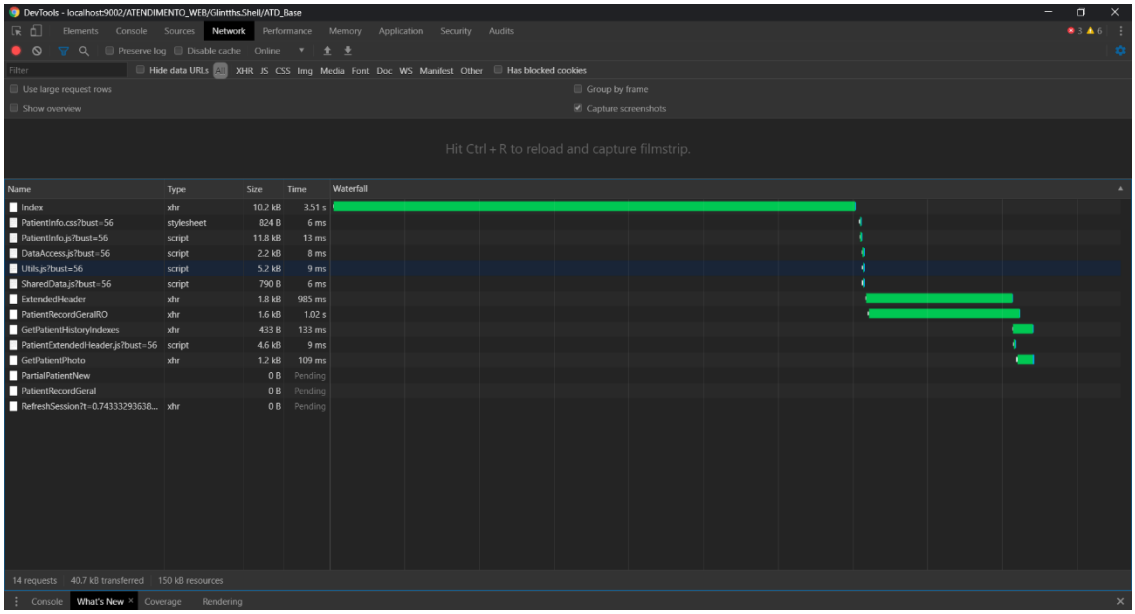


Figura 89 - Ficha de doentes após intervenções

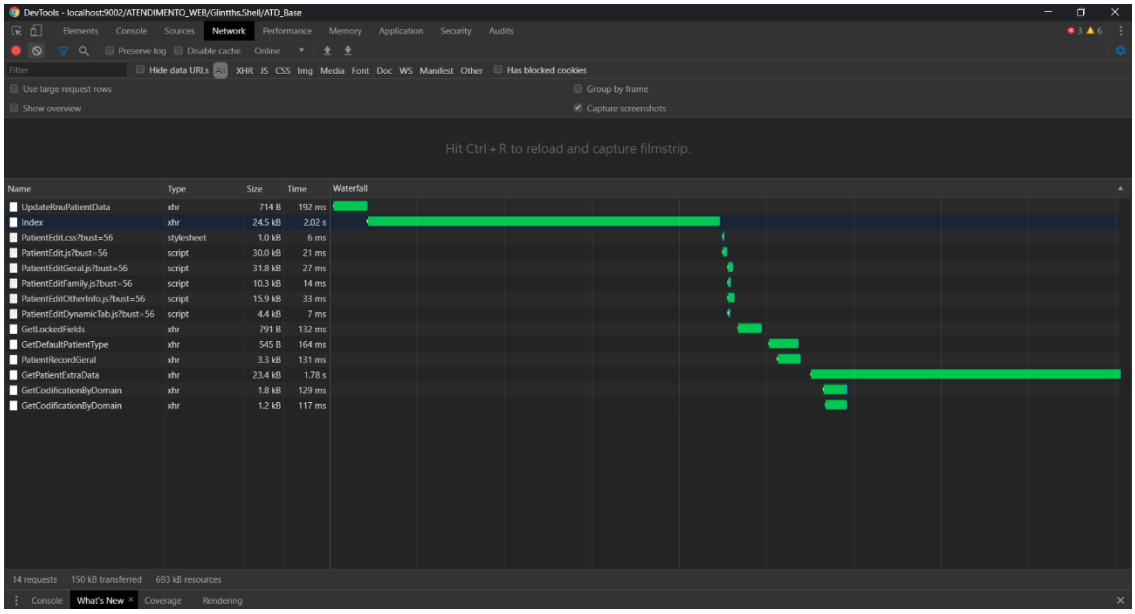


Figura 90 - Edição de doentes após intervenções

Anexo C

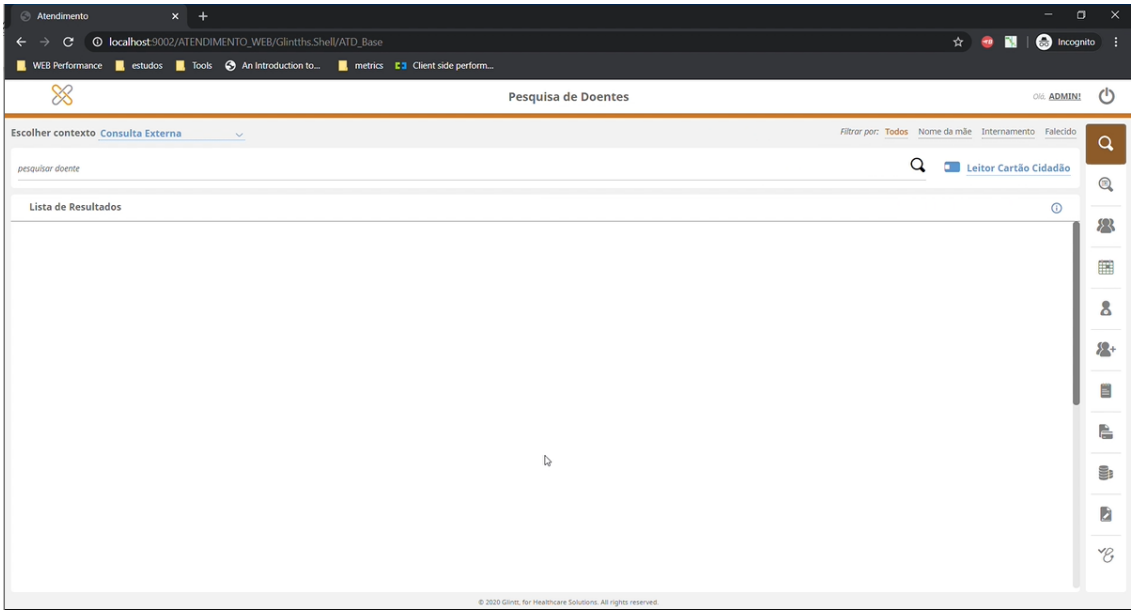


Figura 91 - Página inicial do Atendimento

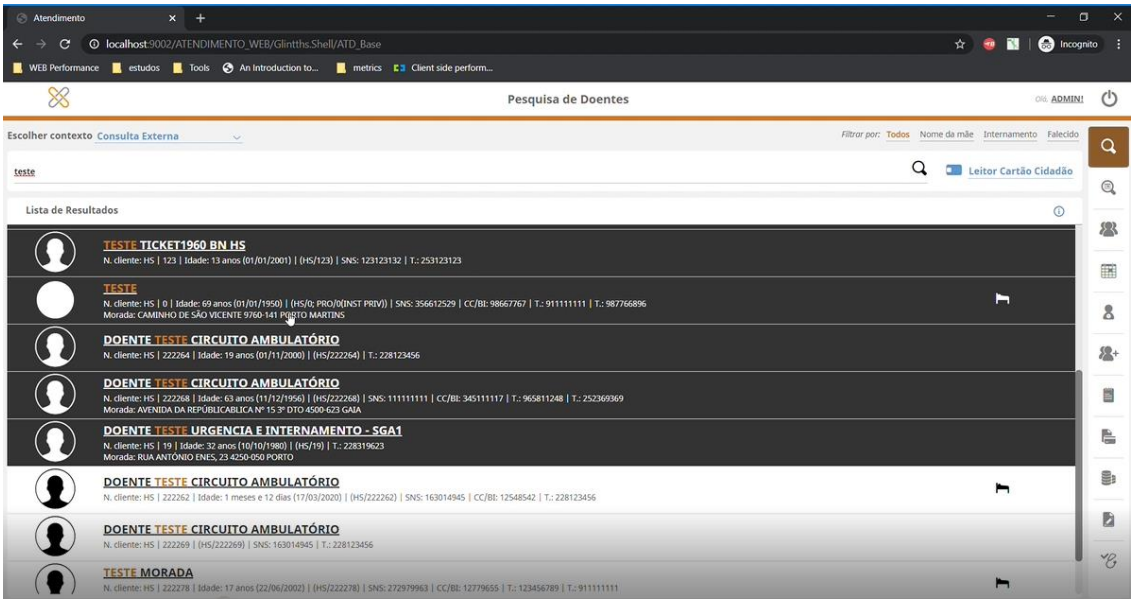


Figura 92 - Pesquisa de doentes

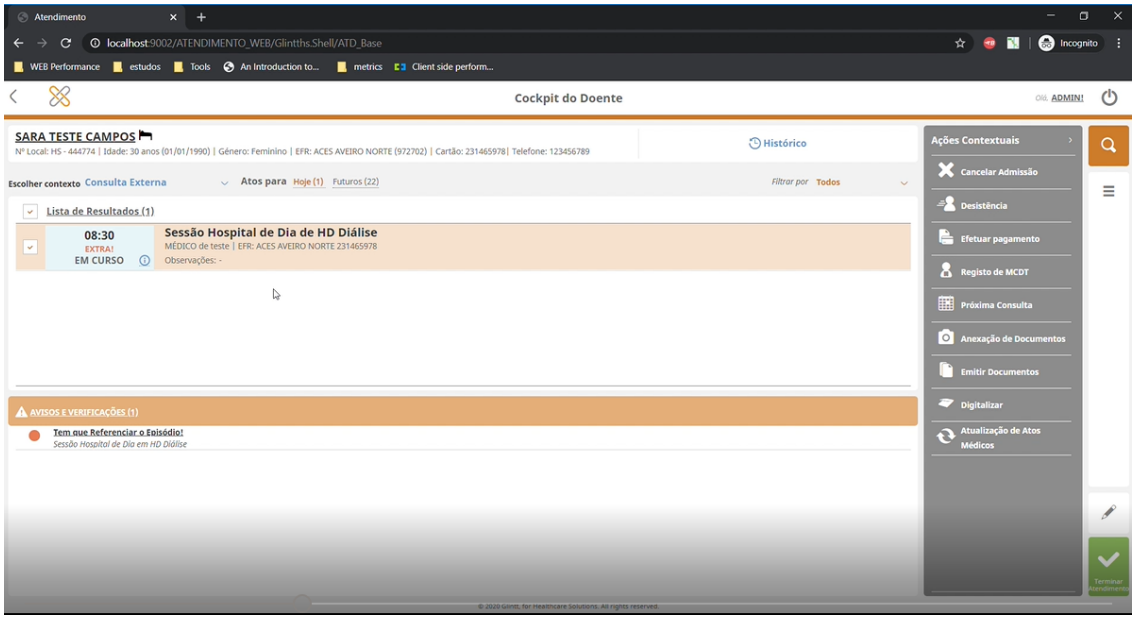


Figura 93 - Cockpit do doente

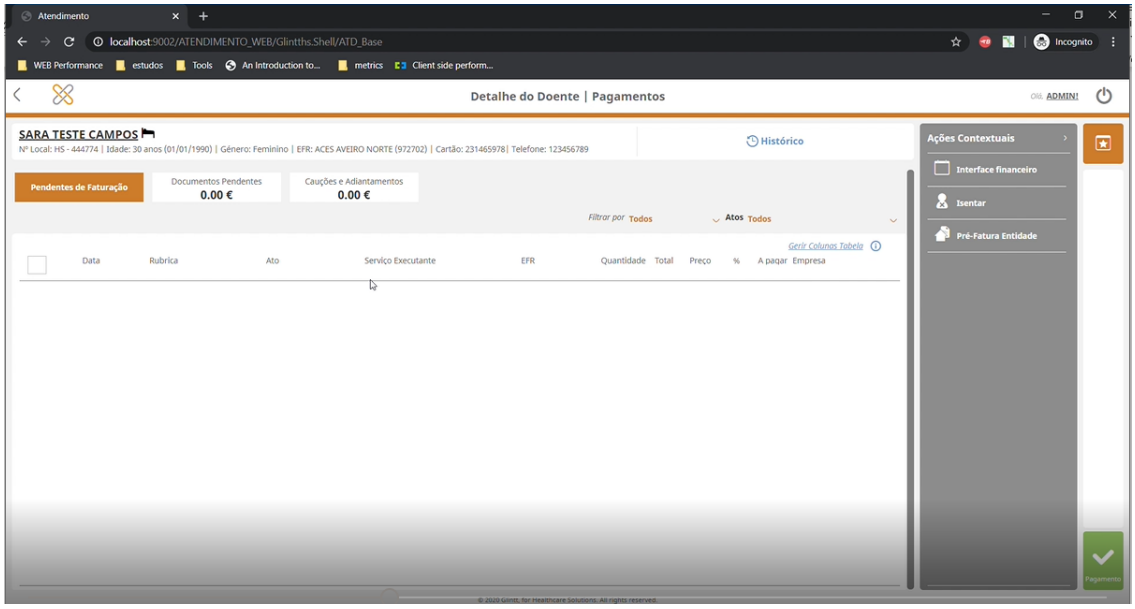


Figura 94 – Pagamento

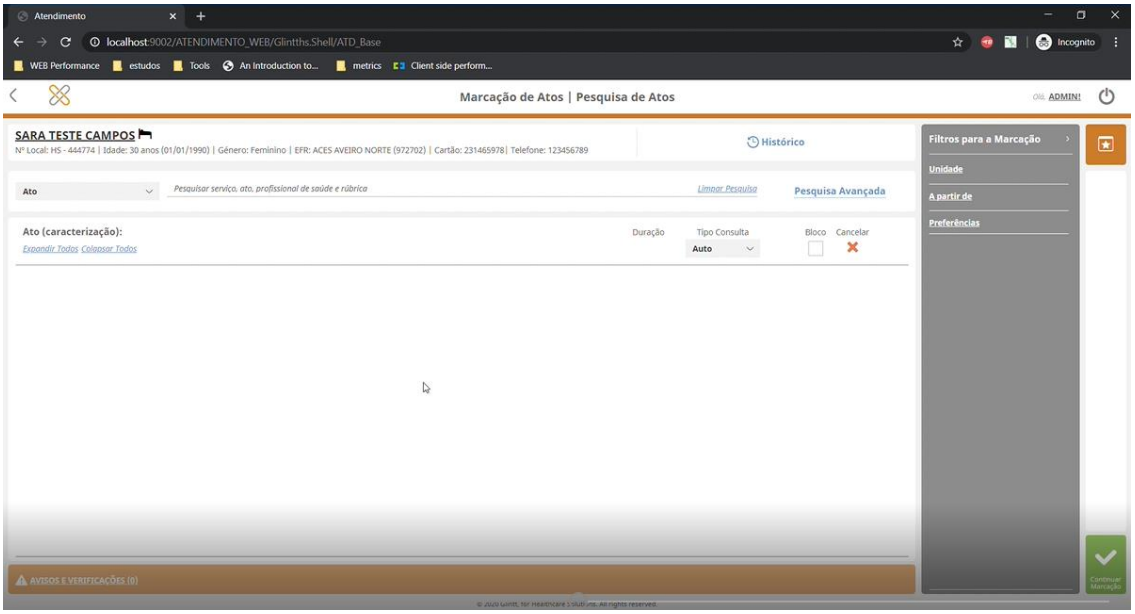


Figura 95 – Marcação

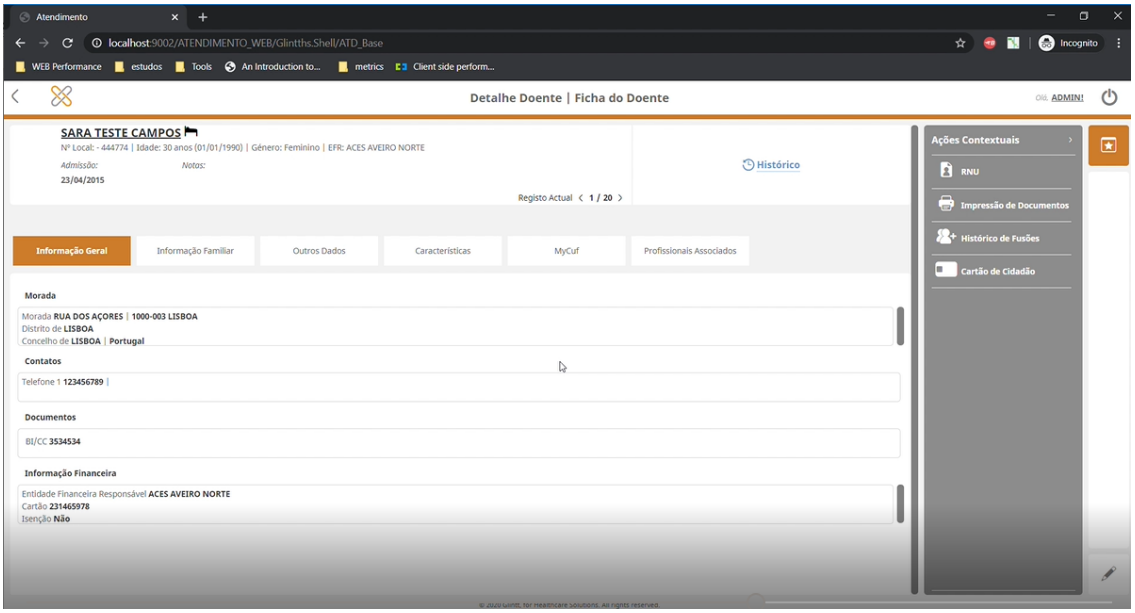


Figura 96 - Ficha de doente

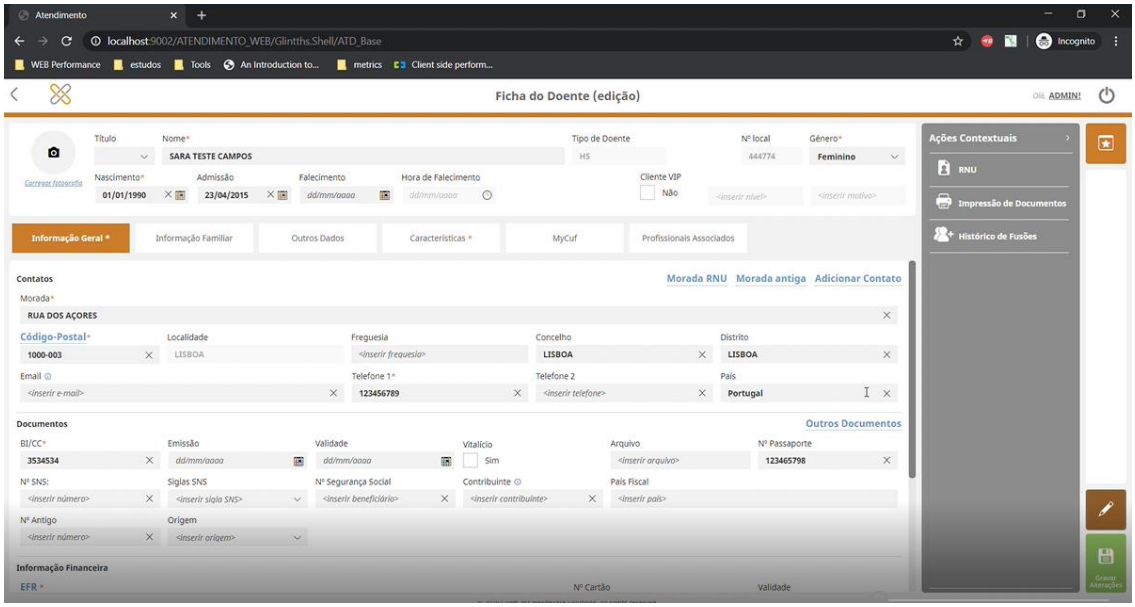


Figura 97 - Edição da ficha de doente

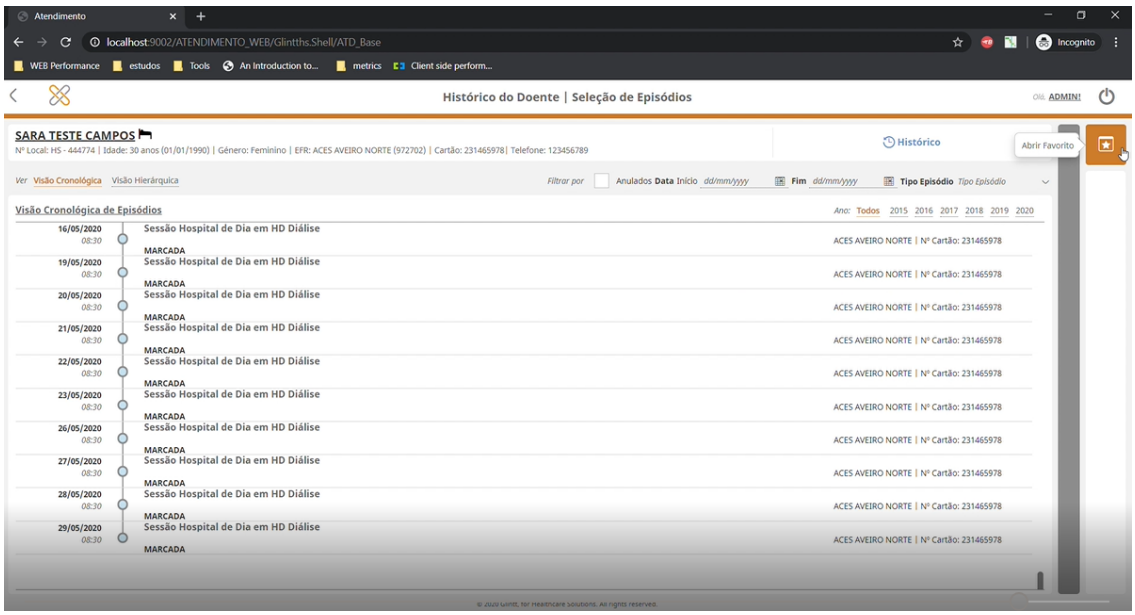


Figura 98 – Histórico

Anexo D

Relatórios Lighthouse

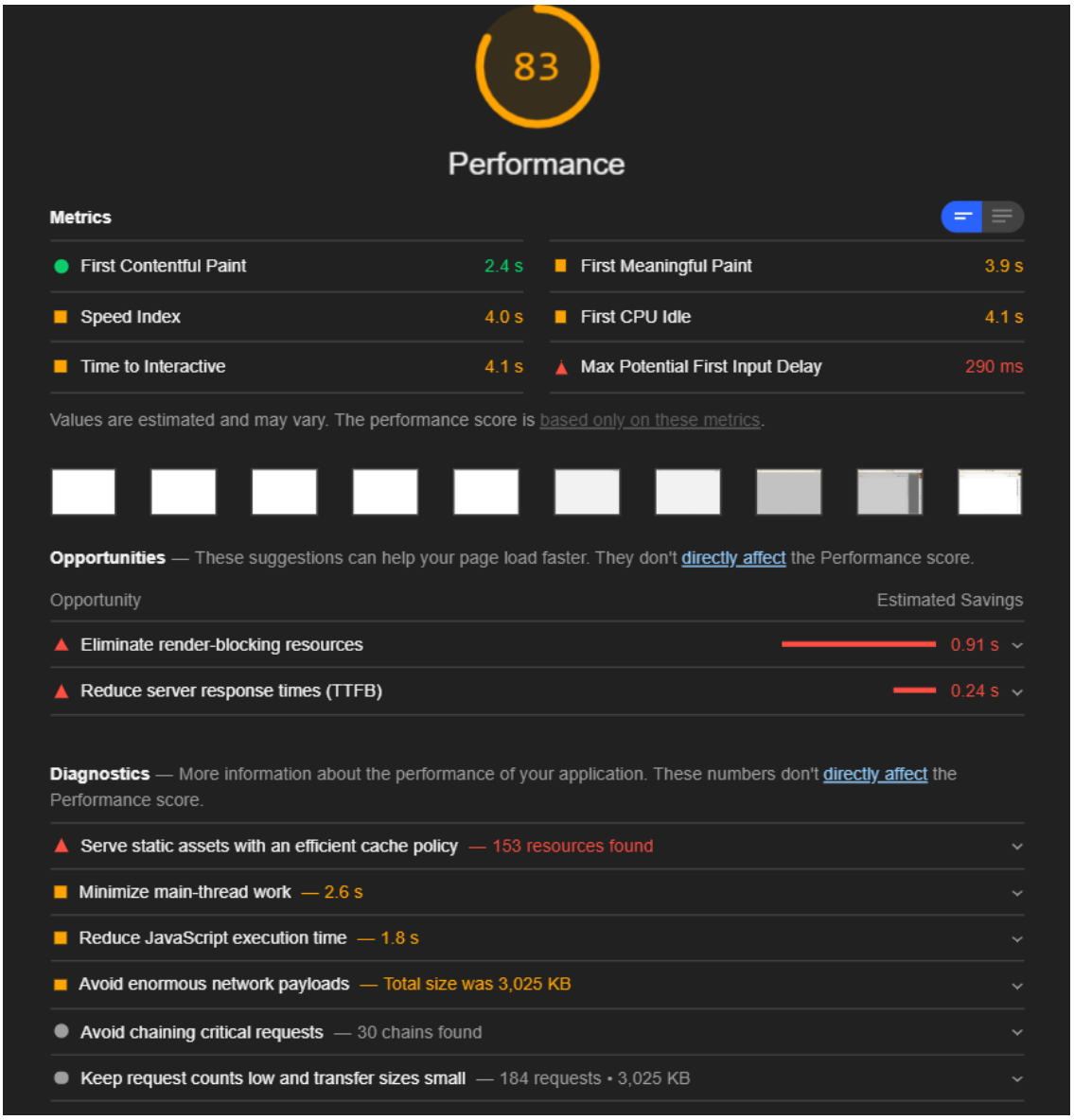


Figura 99 – Relatório Lighthouse antes das alterações

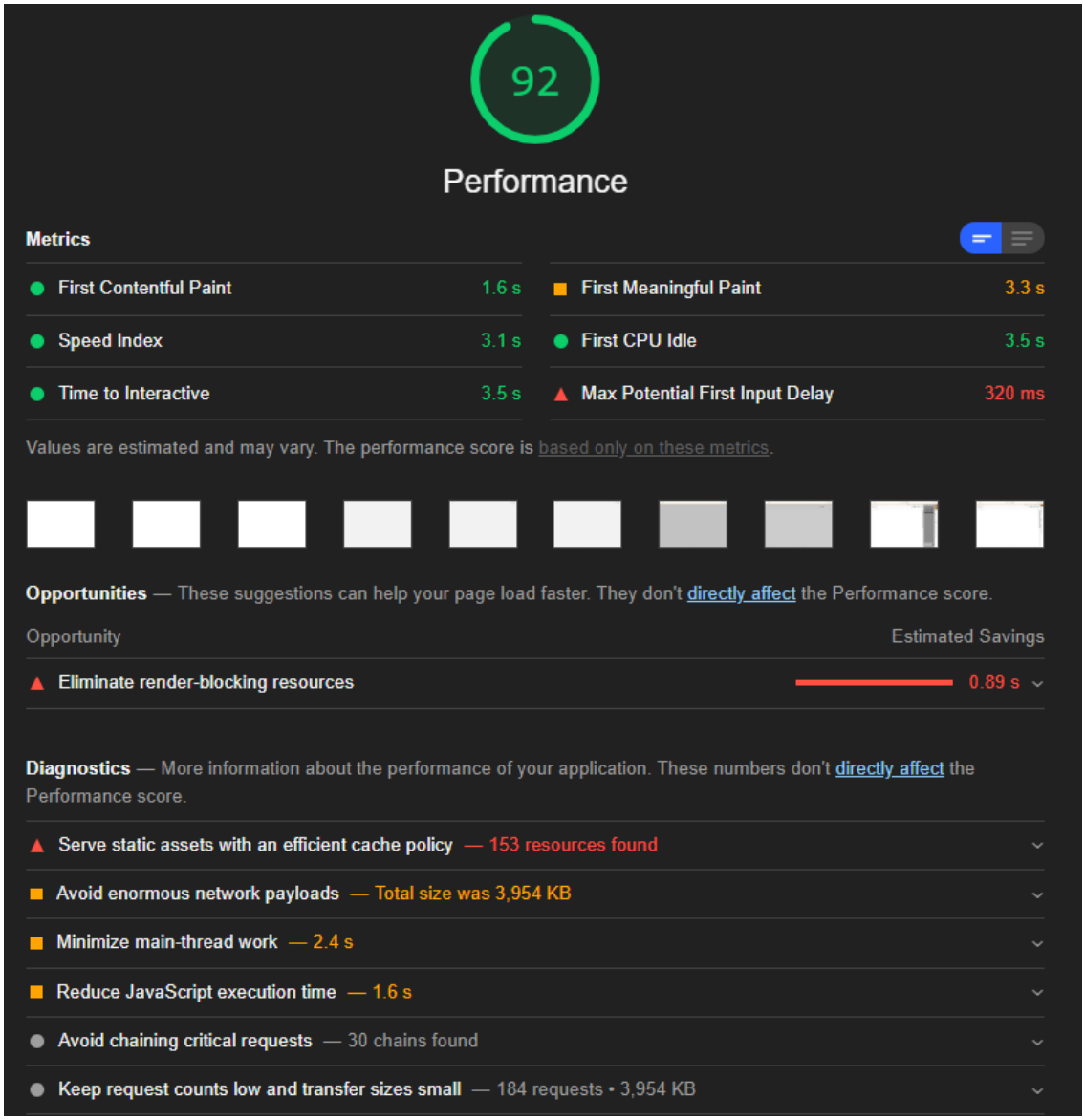


Figura 100 - Relatório Lighthouse após as alterações

Anexo E

Excertos de código

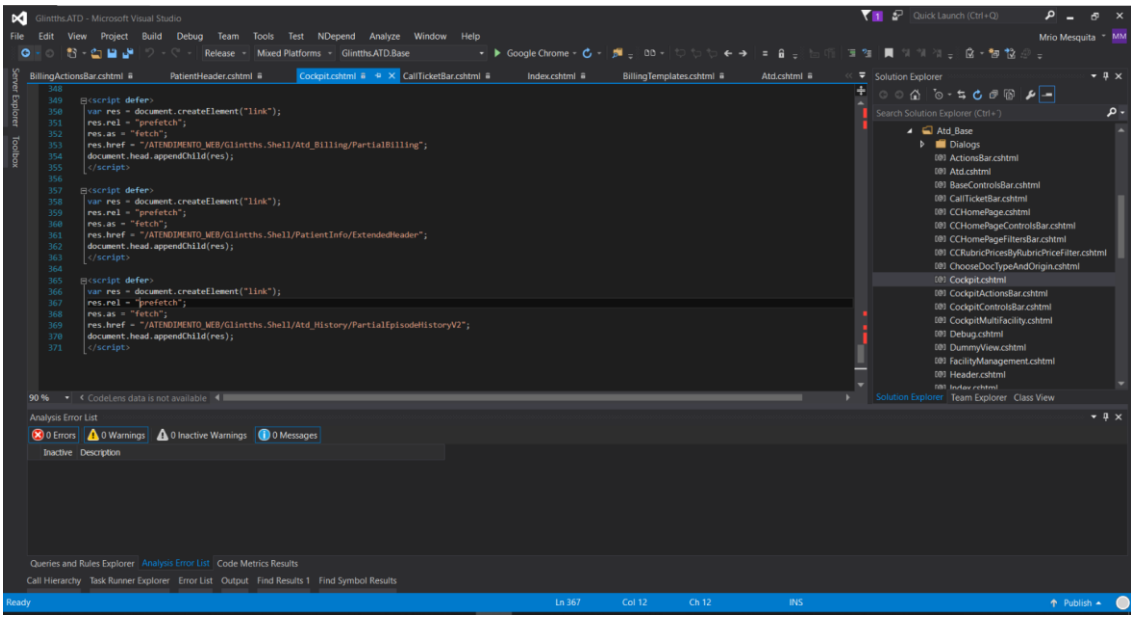


Figura 101 - Excerto de código da pré-procura no Cockpit

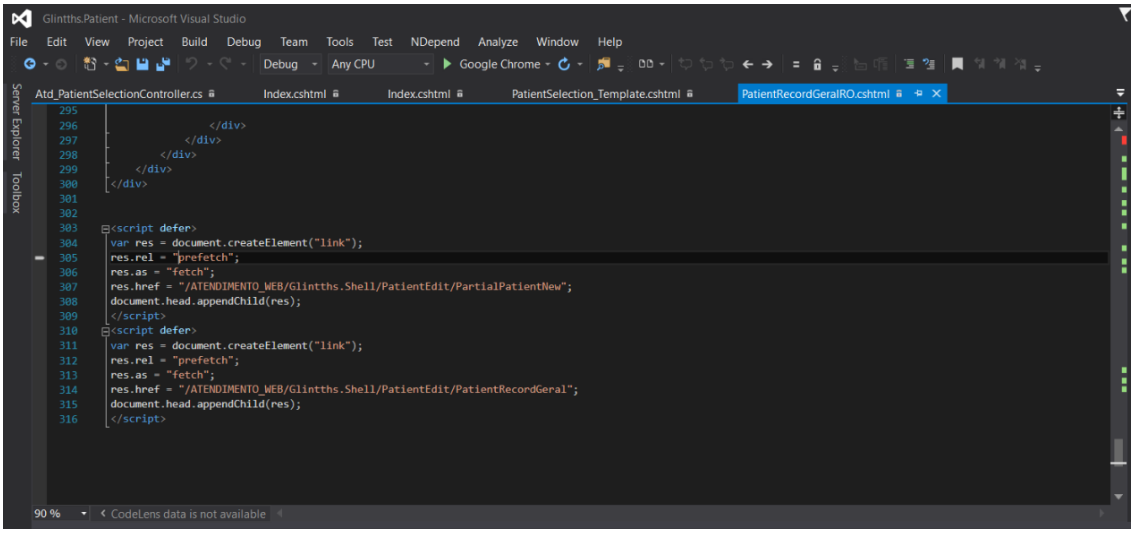


Figura 102 - Excerto de código da pré-procura na ficha de doente