UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

MATHEUS PROLA PFITSCHER

Codestand - Uma plataforma para gerência de novos protocolos do IETF

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Lisandro Zambenedetti Granville

Porto Alegre
2016

"The saddest aspect of life right now
is that science gathers knowledge faster
than society gathers wisdom."
— Isaac Asimov
and me, of course.

AGRADECIMENTOS

É fundamental que o primeiro agradecimento seja feito ao meu pai Estafano Luiz Pfitscher, que permitiu que eu chegasse até este momento dentro da graduação. Na sequência, tenho que agradecer ao Professor Lisandro Granville pela brilhante tutela durante este trabalho de graduação. Preciso também agradecer minha mãe Mara Rúbia e meu irmão Stefano, que inclusive fez a revisão da presente monografia. Outros agradecimentos especiais se fazem necessários, como, agradecer minha namorada Victória Duarte pelo apoio incondicional. Além de alguns amigos que jamais me deixaram me abater, seja durante a graduação ou na realização deste trabalho, são eles, Taís Bellini, Arthur Jacobs e Henrique Lopes.

Em especial, no trabalho do Codestand, preciso agradecer fundamentalmente ao Wanderson e ao Christian pela orientação e esforço durante o longo período de desenvolvimento do projeto. Também agradeço a todos com que trabalhei dentro do laboratório 210 e que permitiram que o tempo que passei lá desenvolvendo o Codestand e a presente monografia fosse tão agradável e vantajoso.

De modo geral quero agradecer a todos que por ventura não mencionei nominalmente mas que fizeram parte da escalada até a conclusão desta monografia. De verdade, muito obrigado!

RESUMO

O IETF é uma organização reconhecida mundialmente pelo desenvolvimento de padrões abertos para a Internet. Com a velocidade que a Internet evolui atualmente, vem se tornando cada vez mais necessária a ampliação da comunidade científica para que o maior número de pessoas queiram implementar e consolidar as novas tecnologias. Porém, o processo de desenvolvimento, em especial dos novos protocolos de rede do IETF, necessita de aprimoramentos. Atualmente, este processo é complexo e demanda tempo demais para que o surgimento de novas ideias possam, finalmente, se tornarem padrões utilizados largamente na Internet.

Para solucionar essa problemática, foi criado o Codestand, uma plataforma destinada a gerenciar o processo de consolidação de novos protocolos para o IETF. Com uma dinâmica semelhante à de uma rede social, esta permite que usuários requisitem implementações para suas propostas teóricas de novos protocolos de rede. Qualquer usuário, seja um antigo membro do IETF ou um estudante que recém ingressou na faculdade, pode implementar a sua própria versão dos protocolos, se estiver interessado. Ao final do processo, o proponente que tiver a ideia de protocolo recebe implementações para validar e identificar erros no seu projeto e o usuário que a implementou recebe gratificação - e visibilidade por contribuir com o IETF. Como resultado, ao final desse processo, a Internet torna-se mais forte e plural.

Palavras-chave: Internet. IETF. Padrões abertos. Protocolo de rede.

ABSTRACT

The IETF is an organization globally recognized for the development of open standards for the Internet. Considering the speed that the Internet evolves today, it is increasingly necessary to expand the scientific community so that more people feel the need to implement and consolidate new technologies. However, the process of development, in particular of the new IETF network protocols, needs improvement. Today, this process is too complex and time consuming to wait for the emergence of new ideas until they finally become widely used standards on the Internet.

To solve that problem, Codestand was created, a platform designed to manage the process of consolidating new protocols for the IETF. Using a social-network-style dynamic, it allows users to request for implementations in their theoretical proposals for new network protocols. Any user, whether a consolidated IETF member or a college freshman, has the ability to implement their own version of the protocols. At the end of the process, the one who had the idea of the protocol receives implementations to validate and identify errors in their project, the user that has implemented them receives gratification - and visibility for contributing with the IETF. As a result, at the end of the process, the Internet becomes stronger and plural.

Keywords: Internet, IETF, Network Protocol, Open Standard.

LISTA DE ABREVIATURAS E SIGLAS

IETF	Internet Engineering Task Force
ISOC	Internet Society
RFC	Request For Comments
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
ARPANET	Advanced Research Projects Agency Network
SQL	Structure Query Language
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
CSS	Cascading Style Sheets

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A rede mundial de computadores (Internet) teve seu início em uma época na qual a educação era estritamente tradicional, realizada através de documentos físicos e papéis. Entretanto, nos últimos vinte anos, a Internet cresceu de forma bastante acelerada e teve seu acesso difundido, de modo que, atualmente, boa parte dos estudantes já possui acesso livre a computadores - o que propicia grande interação entre alunos e professores. Isso significa que as novas formas de aprendizado só tendem a se tornar mais e mais digitais, crescendo e, eventualmente, substituindo as técnicas tradicionais de aprendizado. Nos cursos computacionais, em especial e temos como exemplo a ciência da computação - a Internet é parte fundamental do ensino, atualmente com relevância igual, ou maior, à parte teórica tradicional da matéria. Analisando o progresso histórico nessa área, a tendência é que cada vez mais pessoas passem a depender do funcionamento da Internet para desenvolverem e realizarem suas tarefas. Entretanto, a fim que a Internet possa cumprir esse papel fundamental na sociedade moderna, ela necessita estar em constante evolução, inovando para se tornar cada vez mais eficiente e sustentável - principalmente com relação às novas tecnologias que estão por surgir. Dessa forma, para que a Internet possa seguir acompanhando e promovendo o desenvolvimento da sociedade, é necessário que a comunidade de computação, em especial da área de redes de computadores, se una para desenvolver e dar o suporte necessário para as tecnologias da rede mundial.

Hoje, um dos principais mecanismos responsáveis pelo crescimento da rede mundial de computadores é o *Internet Engineering Task Force* (IETF). Conforme a definição de (ALVESTRAND, 2004), a missão do IETF é aprimorar o funcionamento da Internet através da produção de documentos técnicos que influenciam o modo como as pessoas projetam, usam e gerenciam a Internet. Para alcançar esse objetivo, a organização desenvolve documentos e protocolos que, então, possivelmente se tornarão padrões na área. Esse processo de desenvolvimento dos protocolos se inicia quando alguém propõe um *Internet-Draft* (I-D), uma espécie de esboço da idéia, que, logo, é submetido a comentários, podendo se consolidar e se tornar uma *Request For Comments* (RFC). As RFCs são uma conhecida e consolidada técnica de definição de padrões na Internet. Em resumo, um *Internet-Draft* pode ser submetido por qualquer pessoa, e caberá ao IETF decidir se o documento deve ou não se tornar uma RFC. Ao final, no caso desta receber um interesse considerável, a RFC pode se consolidar e se tornar um padrão na Internet. Atualmente existem mais de 8000 RFCs (RFC-INDEX, 2016), sendo algumas delas parte essencial da

Internet, como por exemplo TCP (DARPA; USC, 1981) e UDP (POSTEL, 1980)

O processo para um *Internet-Draft* se tornar uma RFC, no entanto, pode ser demorado. Isso se deve, principalmente, aos diversos aprimoramentos e refinamentos necessários para que o documento chegue ao chamado *Rough Consensus and Running Code* - um julgamento dos participantes da plataforma combinado com a experiência do mundo real na implementação e implantação das especificações. Além disso, o processo é basicamente realizado através de listas de *e-mail* e encontros presenciais que ocorrem, atualmente, três vezes ao ano. Sendo assim, pode ser uma tarefa um tanto quanto complexa acompanhar o andamento da avaliação, enquanto não se obtém um consenso definitivo sobre a importância do documento. Considerando que o documento proponha um novo protocolo de rede, até que este seja considerado válido, é fundamental que existam pelo menos duas implementações independentes da proposta, e que, dessa forma, sejam explorados o maior número de possíveis problemas. Logo, é de grande importância expandir cada vez mais a comunidade científica, buscando por novos potenciais colaboradores ao IETF. Entretanto, também é sabido que ingressar no *Internet Engineering Task Force* demanda um certo tempo de adaptação, visto que existe uma metodologia de ingresso que demanda bastante tempo disponível. Como consequência, este momento de adaptação por vezes acaba afastando alguns interessados, principalmente os estudantes mais novos que, por ventura, ainda não tenham o devido conhecimento técnico. Assim, é perceptível a necessidade de uma plataforma que permita não só gerenciar o processo de padronização de novos protocolos, como proporcionar um meio de comunicação entre o IETF e potenciais novos contribuidores.

Por meio de um esforço conjunto entre o IETF e o Instituto de Informática da UFRGS, surgiu o Codestand. Codestand é uma plataforma Web que tem como propósito gerenciar e auxiliar no andamento de novos protocolos do IETF. Na interface, ele se assemelha a uma rede social, onde os mentores propõem ideias de novos protocolos, especificando como gostariam que fossem desenvolvidos e, a partir de então, os codificadores interessados implementam e validam o proposto. Portanto, o Codestand busca ser atrativo para que os usuários desejem implementar os protocolos, proporcionando uma maior agilidade e eficiência na consolidação dos padrões. Como resultado final, a plataforma deverá hospedar propostas em diversas linguagens e tecnologias, minimizando as dúvidas sobre a relevância e corretude de cada protocolo. A plataforma deve, também, permitir que os protocolos do IETF sejam facilmente relacionáveis, de modo que os projetos criem uma rede de conhecimento em torno dos protocolos desenvolvidos. Atualmente, o Co-

destand está em produção em *https://codestand.ietf.org*, mais especificamente, em fase de testes com usuários reais e tendo a divulgação da plataforma sendo promovida pelo IETF. Espera-se que, com a plataforma funcionando, ela possa auxiliar na missão de tornar a Internet melhor, mais forte e plural.

O presente trabalho está estruturado da seguinte forma. Na seção 2 é abordado o surgimento e a importância do IETF. Na seção 3 será abordada a parte operacional da plataforma Codestand. Na seção 4 serão tratados os modelos e tecnologias utilizadas na implementação do Codestand. Na seção 5 são abordadas duas visões de avaliação da plataforma. E por fim, na seção 6 será apresentada a conclusão e os trabalhos futuros relacionados a este projeto.

2 IETF

Desde a sua origem, o IETF cresceu em uma grande e larga comunidade internacional de desenvolvedores de rede, operadores, fabricantes e pesquisadores preocupados com a evolução da arquitetura da Internet e o bom funcionamento da operação da Internet (HOFFMAN; BRADNER, 2002).

2.1 Contextualização histórica

O *Internet Engineering Task Force* (IETF)) é uma organização de padronização reconhecida mundialmente como um dos elos fortes da Internet, hoje responsável por desenvolver boa parte dos padrões e protocolos que vigoram na Web. Surgido em 1986, no início, tratava-se apenas de um fórum para coordenação técnica, contratada pela *Defense Advanced Research Projects Agency* (DARPA) para trabalhar na ARPANET (FROEHLICH; KENT, 1991), na *US Defense Data Network* (PIKE, 2011) e nos sistemas de *gateway* (porta de entrada) do núcleo da Internet. A partir de 1993, o IETF deixou de ser uma comunidade administrada e incentivada pelo governo dos Estados Unidos e passou aos cuidados da *Internet Society*, uma organização internacional sem fins lucrativos que promove a expansão da Internet. A referida organização atua dando apoio financeiro e jurídico para grupos que tenham o mesmo objetivo, qual seja, proporcionar o crescimento da Internet, dando suporte em particular ao *Internet Engineering Task Force*. Ao separar-se do governo dos Estados Unidos, o IETF iniciou uma expansão global, possibilitando que pesquisadores, fabricantes e entusiastas do mundo inteiro pudessem aderir à organização. Além disso, como consequência dessa separação, o IETF deixou de participar de projetos específicos para se dedicar a estabelecer padrões para a Internet, auxiliando e indicando boas práticas na área de redes de computadores, ou seja, como devem ser desenvolvidas e gerenciadas as redes.

O alcance do IETF e sua relevância no meio com certeza evoluíram muito ao longo dos anos. Na sua primeira reunião em 1986, na cidade de San Diego, nos Estados Unidos, compareceram nada mais do que vinte e uma pessoas (IETF-1, 1986). Porém, recentemente, no IETF-94 (IETF-94, 2015), que ocorreu em novembro de 2015 em Yokohama, no Japão, o número de participantes, remotos ou presenciais, já era de 1.487 pessoas. Tamanha expansão demonstra uma grande evolução do IETF, sendo importante destacar que estes números indicam apenas os membros que puderam comparecer no evento, sem

considerar os tantos outros que colaboram ou já contribuíram de alguma maneira com a comunidade em seus trinta anos de existência. Portanto, não seria difícil estabelecer um paralelo entre o crescimento do IETF e o avanço da Internet, claramente concomitantes.

Mesmo com alguns números bem significativos, e avanços tecnológicos que podem ser percebidos por todo o mundo, a organização ainda demonstra uma grande preocupação em captar cada vez mais pessoas ao redor do mundo. Um exemplo disto foi a IETF-95 (IETF-95, 2016), que aconteceu em março de 2016, em Buenos Aires, Argentina, a primeira de seu tipo na América do Sul. Nesta reunião, o total de participantes da América do Sul foi de 140 - o que, comparativamente aos partícipes da Europa e da América do Norte não é muito, mas já é um recorde em relação a reuniões anteriores. Espera-se que, com o passar das reuniões, a participação latino-americana cresça cada vez mais, possibilitando uma maior integração entre os continentes. Afinal, não faz muito sentido que as pesquisas sobre a Internet se restrinjam ao hemisfério norte, pois, sob o ponto de vista do IETF, quanto mais gente houver para opinar e contribuir, mais plural, justa e forte a Internet tende a ser no futuro. Até porque hoje é difícil imaginar uma ferramenta mais global que a Internet e que proporcione tantos avanços em tantos setores.

2.2 Objetivos e princípios

A história da Internet está diretamente ligada a *Internet Engineering Task Force*, uma vez que praticamente todos os padrões utilizados na Internet hoje foram desenvolvidos sob seus cuidados. Mas, é claro que, assim como a Internet, a organização está em constante crescimento e muitas mudanças ocorreram ao longo do tempo, porém, mesmo assim, os objetivos da organização se mantiveram e proporcionaram que o foco continuasse em fazer a Internet funcionar cada vez melhor.

2.2.1 Missão

Segundo (ALVESTRAND, 2004) a missão do IETF é desenvolver padrões para a Internet e, para que seja bem sucedida, ele a divide como:

- Identificar e propor soluções para problemas operacionais e técnicos na Internet.
- Especificar o desenvolvimento ou uso dos protocolos, além da arquitetura, para resolver tais problemas técnicos na Internet.

- Fazer recomendações sobre a padronização de protocolos e o modo de utilização dos mesmos na Internet.
- Facilitar a transferência de tecnologia dos grupos de pesquisa para toda a comunidade.
- Prover um fórum para trocar informações com a comunidade da Internet, entre eles, fabricantes, usuários, pesquisadores, prestadores de serviço e gerentes de rede.

Diante da compreensão desta missão, torna-se mais acessível o entendimento acerca do trabalho que vem sendo desenvolvido ao longo dos anos. Pois bem, o IETF não apenas está preocupado em preparar uma Internet para o futuro, mas em solucionar os problemas atuais. Tendo como principais objetivos disseminar o conhecimento, a organização procura prover meios de comunicação entre a comunidade, além de elaborar padrões abertos, para que todos possam usufruir e ajudar na implementação e implantação de novas tecnologias. Importante destacar que a terminologia Padrões Abertos pode variar de acordo com a organização padronizadora, que neste caso é o IETF. Seguindo a definição, desenvolver Padrões Abertos tem o objetivo de facilitar e ajudar no desenvolvimento de novas tecnologias. Porém, isso não significa que serão sempre padrões gratuitos. O IETF permite que seus padrões possam conter especificações cuja implementação acarretará no pagamento de taxas de licenciamento de patentes, seguindo a *Reasonable and non-discriminatory terms* (Condições razoáveis e não discriminatórias) (RAND, 2013).

2.2.2 Paradigmas

Uma vez entendida a missão, é essencial compreender também os princípios que, segundo (ALVESTRAND, 2004) visam assegurar que o objetivo estipulado será perseguido e respeitado:

- Processo aberto - Em que qualquer pessoa interessada pode participar, desde que saiba exatamente o que está sendo decidido e que, posteriormente, esforce-se ao máximo para *"make his or her voice heard"* (Fazer a sua voz ser ouvida). Além disso, as minutas das reuniões devem ser mantidas totalmente públicas na Internet.
- Competência técnica - Para argumentar nas questões nas quais o IETF produz seus documentos, o participante deve ter competência para falar a respeito do assunto e estar disposto a ouvir qualquer informação de fonte competente. Por conseguinte, é esperado que os resultados gerados sejam de alta qualidade, o que é conhecido no

ramo como ”Qualidade de engenharia”.

- Núcleo voluntário - Os participantes e as lideranças devem participar do IETF apenas pelo total interesse em ajudar na missão de fazer a Internet melhor e mais forte.
- *Rough consensus and Running code* (Amplo consenso e Código rodando) - Implica em criar padrões baseados na combinação do julgamento dos participantes e na experiência de mundo real com a implementação e implantação das especificações.
- Propriedade do protocolo - Quando o IETF obtém a propriedade de um protocolo ou função, é aceita a responsabilidade por todos os aspectos do protocolo, mesmo que alguns aspectos possam nunca ser visto na Internet. Eventualmente o IETF pode não ser responsável pelo protocolo ou função, logo, não tentará exercer qualquer controle sobre ele, mesmo que às vezes este possa afetar a Internet.

Analisando os princípios do IETF, é possível perceber grande destaque para uma produção científica concretizada, através de um processo aberto e democrático, no qual a opinião de todos deve ser respeitada. Diretrizes como Processo Aberto e Voluntariado, refletem a razão da existência da organização: produzir padrões abertos, de modo que toda a comunidade se beneficie. Quando combinado com Competência Técnica, espera-se que, além de abertos, os padrões sejam realmente confiáveis e que, na medida do possível, sejam o melhor que poderia ter sido desenvolvido naquele momento. Já a Propriedade do Protocolo é sinalizada como uma medida de proteção, assumindo que todas as complicações acerca do protocolo são de responsabilidade do IETF. Certamente, no entanto, a diretriz que merece o maior destaque é o *Rough Consensus and Running Code*, uma das chaves do modo de operação do IETF.

A primeira parte da diretriz, o *Rough Consensus*, indica que as decisões devem ser tomadas de forma conjunta, ou seja, o consenso deve prevalecer em relação a opiniões individuais. Essa diretriz mostra a importância de ter o maior número de pessoas de diferentes países envolvidas, pois assim as decisões tomadas serão cada vez mais um consenso, ao invés de imposição de alguns poucos. Na prática, este mecanismo consiste não na obrigatoriedade de unanimidade sobre as propostas, mas sim na aceitação de uma ampla maioria. Por exemplo, geralmente se a proposta tiver mais de 90% de aprovação, ela provavelmente será aceita, e se compreender menos de 80%, tende a ser rejeitada (BRADNER, 1999).

O *Running Code*, por sua vez, estipula que são necessárias pelo menos duas implementações independentes da proposta para ela ser avaliada, pois, desta forma, erros e falhas podem ser mais facilmente percebidos. Ou seja, a adoção ou não de uma proposta pelo IETF depende de demonstrar aos membros que o protocolo funciona na prática.

Assim, mais uma vez, é possível ver a necessidade de ampliar a comunidade, buscando equilíbrio entre bons tutores teóricos e entusiastas do desenvolvimento. Isto é relevante pois o IETF define padrões consultando seus membros e que estes, por sua vez, devem concordar em aceitar ou rejeitar a proposta, admitindo que é a coisa certa a ser feita, usando de motivos técnicos e teóricos. É claro que, se para cada proposta todos os membros devessem refletir, opinar e consentir, o processo levaria ainda mais tempo. Por isso o IETF possui uma hierarquia baseada em subdivisões de grupos de trabalho, onde é dado um enfoque a determinados temas, permitindo que as discussões e consensos sejam realizados por pessoas que tenham total interesse sobre o assunto. Portanto, como podemos ver, é possível afirmar que a Internet só tende a evolui quando tem a teoria e a prática caminhando na mesma direção.

2.3 Estrutura

Se há algo a se destacar a respeito do IETF, em relação às outras instituições de padronização, é a maneira como ela é organizada.

2.3.1 Organização

A organização do IETF existe da seguinte maneira. Começando pelo topo, temos uma área diretora, encarregado das funcionalidades administrativas do IETF. Conhecido como *Internet Engineering Steering Group* (IESG)), este é responsável pelas questões técnicas de gerência do IETF e pela manutenção do processo de padronização da Internet. O corpo que compõe o IESG é eleito por um comitê especializado, com mandato de dois anos de duração (BRADNER, 1999). Dentre as funções que executa, está a aprovação final das especificações desenvolvidas, efetivando-as como padrões da Internet, além da aprovação de novos grupos de trabalho. Os grupos de trabalho são um mecanismo que permite a divisão do trabalho em grupos de interesse, de modo a permitir que os membros sejam pessoas altamente interessadas naquele tópico. Os grupos de trabalho são tipicamente criados para resolver um problema específico, ou produzir um ou mais produtos específicos - por exemplo, uma diretriz ou uma especificação de normas. Normalmente é esperado que eles tenham vida curta, ou seja, que existam até que completem suas metas, quando então são terminados.

Apesar da dinâmica de *Rough Consensus*, que permite que todos tenham voz ativa, por motivos de ordem, cada grupo de trabalho possui usualmente dois *Chairs* (Presidentes). Os *Chairs* são responsáveis por realizarem as funções administrativas do grupo, além de exercer uma espécie de liderança. Grupos de trabalho normalmente levam suas agendas de modo simultâneo,já que o IETF não pode concentrar-se apenas em projetos específicos. Atualmente, existem cerca de 115 grupos de trabalho dentro da organização, divididos em oito áreas diferentes (BRADNER, 1999). Na Tabela 2.1 podem ser vistas os nomes das áreas e as descrições de cada uma, dando uma noção básica de como os membros se agrupam para resolver os problemas. Assim como os grupos de trabalho, cada área possui um diretor, que é o responsável por manter as diretrizes da área alinhadas. Cada área tende a se dividir em diversos grupos de trabalho, mas o agrupamento existe principalmente para tentar estabelecer uma correlação entre os objetivos.

Tabela 2.1: Áreas do IETF

Área	*Descrição*
Aplicações (APP)	Protocolos vistos por programas de usuários, como *e-mail* e a *web*
Geral (GEN)	Engloba em seus grupos de trabalho tudo que não se encaixa em outras áreas (que é muito pouco)
Internet (INT)	Diferentes formas de transferir pacotes IP e informações de DNS
Operações e Gerenciamento (OPS)	Aspectos operacionais, de rede e configuração
Aplicações de tempo real e Infraestrutura (RAI)	Comunicações pessoais sensíveis a atrasos
Roteamento (RTG)	Levando pacotes aos seus destinos
Segurança (SEC)	Autenticação e privacidade
Transporte (TSV)	Serviços especiais para pacotes especiais

Fonte: (HOFFMAN; HARRIS, 2006)

Devido a essa divisão bem estabelecida de hierarquia, as decisões são tomadas por quem está interessado nos assuntos que concernem àquela área e, mais especificamente, que dizem respeito ao determinado grupo de trabalho. Portanto, a estrutura é constituída de forma *bottom-up* (de baixo para cima), com grupos de trabalho formados por pequenos grupos de interessados, que se reúnem por conta própria e, em seguida, propõe o grupo de trabalho para o diretor de alguma área que, por fim, será aprovado ou não pelo IESG.

2.3.2 Elaboração dos padrões

O processo de desenvolvimento dos padrões dentro dos grupos de trabalho possui uma forma bem definida, mas para entender este processo, é preciso compreender alguns conceitos fundamentais. O IETF produz padrões para a Internet na forma de documentos que possuem formatos específicos, e o entendimento de como estes funcionam é indispensável para a compreensão do processo como um todo. Neste ambiente, qualquer pessoa interessada pode submeter um documento, denominado *Internet-Draft* (I-D) (Rascunho). Neste documento devem estar contidas as especificações preliminares, resultados de pesquisas relacionadas à proposta, dentre outras informações técnicas. Caso de que seja considerado promissor, o *Internet-Draft* possivelmente será adotado por um grupo de trabalho que, por regra, é englobado por uma das oito áreas. Logo, este documento ficará disponível para comentários dos leitores e, conforme os questionamentos, os autores podem incorporar o que for conveniente ao proposto. Então, o documento passa a ser diretamente submetido à política do *Rough Consensus and Running Code* e, após a sabatina de comentários e constantes reformulações, ocorridas após um tempo determinado, é decidido se deve-se manter ou recusar a proposta. A fim de que esse documento se torne um Padrão, o mesmo deve então evoluir em um *Request for Comments* (RFC).

Uma RFC é uma espécie de memorando que descreve métodos, comportamentos, pesquisas ou inovações aplicáveis para o funcionamento da Internet e dos sistemas que se conectam a Internet. Normalmente possuirá termos como *MUST* (Deve) ou *NOT RECOMMENDED* (Não recomendado) de modo a realmente estabelecer as boas e más práticas sobre o assunto em questão (BRADNER, 1997). Para um *Internet-Draft* chegar no estado de RFC significa que ele foi amplamente discutido, testado e validado e que se tem certeza sobre a relevância do mesmo. Por fim, no caso da RFC ganhar interesse suficiente da comunidade, esta pode evoluir em um padrão da Internet.

2.3.3 Dinâmica de trabalho

A mecânica de desenvolvimento dos documentos dentro dos grupos de trabalho e do IETF em si, atualmente consiste basicamente em duas frentes: listas de discussão (via *e-mail*) e encontros presenciais. Como os encontros presenciais acontecem, atualmente, só três vezes ao ano, a maior parte do trabalho é realizada através das listas de discussão. Em geral, as reuniões, que até o presente momento já foram 97, possuem um propósito

mais geral, introduzindo novos grupos de trabalho e apresentando o que fora desenvolvido desde a última reunião. Além disso, esses encontros tem como objetivo permitir que estudantes, pesquisadores e fabricantes da indústria troquem experiências e conhecimento pessoalmente, apresentando a comunidade para ela própria, uma das grandes vantagens de integrar o IETF.

2.4 Plataformas

Visto que a maior parte do processo é feita remotamente e que, de acordo com as diretrizes que regem o IETF, o conteúdo desenvolvido deve ficar disponível para a comunidade, é indispensável a existência de uma ferramenta que auxilie nesta operação.

2.4.1 Datatracker

Para facilitar esse processo, foi criado, em 2001, o Datatracker (DATATRACKER, 2016), uma interface para o dia-a-dia das pessoas que trabalham com IETF. Nele, são disponibilizados todos os documentos, tanto os que estão sob desenvolvimento quanto os que já são considerados padrões na área. Também é possível encontrar no Datatracker, informações a respeito dos grupos de trabalho, próximas reuniões, submeter novas propostas de *Internet-Draft* e tantas outras coisas que concernem o IETF. Seguindo os princípios da organização, o Datatracker é uma solução de código aberto, disponível no SVN (Ferramenta de versionamento de código) e que foi desenvolvida utilizando o *Framework Web* conhecido como Django.

Dado isso, o Datatracker é considerado uma ferramenta extremamente útil ao que se propõe: auxiliar o trabalho diário dos participantes do IETF e armazenar os padrões estabelecidos na área para que toda a comunidade tenha acesso. Ou seja, é uma ferramenta fundamental para as pessoas que precisam de auxílio - buscando os padrões já consolidados - ou para quem já é membro da organização e está desenvolvendo algum documento técnico.

Porém, como já foi visto, o IETF teve um crescimento enorme desde sua fundação em 1986, sentindo hoje, cada vez mais a necessidade de penetrar na comunidade acadêmica e científica. No entanto, especificamente para essa tarefa de captar novos usuários - principalmente desenvolvedores - o Datatracker é falho, uma vez que tem a interface

gráfica e o funcionamento altamente direcionados aos já familiarizados com a plataforma.

2.4.2 Uma nova plataforma

Mesmo sabendo que o IETF é uma comunidade aberta, é também sabido - e reconhecido até mesmo por seus membros - que, para adentrar de fato na comunidade, tendo participação efetiva, é necessária certa experiência e bastante tempo disponível. Isso se dá, pois o processo de elaboração dos padrões é lento e complexo, muito por conta de ser realizado basicamente através das longas listas discussão. Mais que isso, o próprio IETF considera extremamente recomendável o comparecimento de seus membros nos encontros presenciais, tarefa que demanda dinheiro e disponibilidade. E, pra completar, o Datatracker e as metodologias atuais não permitem uma fácil inserção para iniciantes. Um exemplo que ilustra esta problemática são os alunos de graduação, que podem não possuir ainda o conhecimento técnico pleno, mesmo dispondo de grande interesse na área.

Em suma, tanto a lentidão quanto a natureza hermética do processo contrastam com a necessidade da organização de atrair o maior número de interessados em desenvolver os protocolos propostos e testá-los. Isto é tão evidente, que o *Internet Engineering Task Force* e a *Internet Society* já tem demonstrando amplo interesse, e até necessidade, em construir uma plataforma que permita agilizar seu processo de implementação.

Preferencialmente, essa plataforma deveria agregar as vantagens do Datatracker, e, por sua vez, possibilitar também que não-membros do IETF possam contribuir sem dificuldade. Com isso, espera-se que, a medida que um usuário não membro do IETF possa participar de projetos de implementação, ele desenvolva interesse na área e, futuramente, possa vir a participar ativamente da comunidade - seja escrevendo *Internet-Drafts/RFCs*, dando sua opinião nas listas ou participando das reuniões presenciais.

Com a introdução desta nova plataforma, a organização se beneficiaria tanto por expandir sua comunidade quanto por agilizar seu processo. Ao mesmo tempo, estes novos envolvidos seriam beneficiados pelo reconhecimento de participar e de contribuir com o IETF, reforçando o mantra da organização em tornar a Internet cada vez mais forte.

3 CODESTAND

O Codestand é uma plataforma Web (CODESTAND, 2016), no estilo de uma rede social, que tem como intuito melhor gerenciar o processo de elaboração de protocolos do IETF. A solução surge para cobrir algumas lacunas deixadas pelo sistema Datatracker, muitas vezes considerada pouco atrativa para novos membros interessados apenas em implementar protocolos e não necessariamente em escrever novos documentos como *Internet-Drafts* ou RFCs. Isto se dá, pois, o Datatracker não foi inicialmente pensado e desenvolvido com este propósito. Desta maneira, fica difícil encontrar motivos para tentar adaptá-lo às necessidades expostas acima, pois todo seu funcionamento já é voltado a outros fins.

A plataforma descrita neste trabalho de conclusão tem como principal meta apresentar uma interface que seja mais atraente aos usuários leigos, tanto no IETF, quanto aos usuários frequentes do Datatracker. Um exemplo claro são algumas técnicas de desenvolvimento planejadas para serem aplicadas ao Codestand, como gamificação - proporcionar recompensas pelo uso da plataforma - o que demonstra interesse em criar um contraste com o ambiente quase sempre formal do IETF.

Para melhor compreender as formas como o Codestand planeja atuar e cativar seus usuários, são destacados alguns dos ideais que estão descritos e fixados na própria plataforma:

- Ajudar e contribuir para a evolução da Internet - Como o Codestand provém das necessidades do IETF, faz sentido que o mote central de todas as ferramentas criadas pela organização tenham como objetivo final tornar a Internet cada vez mais forte. No caso em específico do Codestand, ainda cabe ressaltar que a meta é tornar também a Internet mais plural.
- Se envolver em trabalhos do IETF e do desenvolvimento de padrões abertos Este item também serve para reafirmar a missão do IETF na medida de que é necessário expandir a comunidade científica e aproximar, ao máximo, mentes que podem transformar a Internet.
- Fácil acesso para implementações de código aberto - Se um usuário um dia tiver sido beneficiado por uso de algum padrão/implementação aberto, é provável que, no futuro, ele queira retribuir. Deste modo, quem ganha é a comunidade científica em geral.

Agora, mesmo sabendo que alguns destes tópicos provavelmente já tenham sido repetidos a exaustão, antes como metas do IETF e agora do Codestand, e mais, que, provavelmente, já tenham sido compreendidos, é necessário atentar para o modo como isto é traduzido na plataforma. Pois, o Codestand é uma plataforma de ideias e implementações, mas, não é, de forma alguma, um ambiente de desenvolvimento, não sendo possível codificar em qualquer linguagem. Tudo que é armazenado na plataforma Codestand são *links* para sites de versionamento de código e para os *Internet-Drafts* e RFCs dispostas no Datatracker. Ou seja, trata-se apenas de uma ferramenta de gerenciamento e aceleração do desenvolvimento dos protocolos do IETF, mas que busca transformar os ideais anteriormente descritos em realidade através da parceria com outras plataformas melhores, e mais consolidadas para os fins específicos.

Outro detalhe importante de destacar é que, por mais que o Codestand tenha surgido num período bastante recente, para cobrir a necessidade latente de gerenciar o processo de desenvolvimento dos protocolos do IETF, isto não obriga a plataforma a descartar o meio como vinha sendo feito através dos anos. E, claro, os que estão em desenvolvimento neste momento ao redor do mundo também podem ser beneficiados.

Para cumprir este fim, não descartar o passado e o presente da atual plataforma, o Codestand dá suporte para o cadastro de projetos de forma retroativa, de modo que é possível documentar projetos em andamento ou já finalizados. Uma vez que tem como diretriz o fácil acesso a implementações de código aberto, é de total serventia que sejam inseridos na plataforma também os projetos bem consolidados. Apesar de não servirem para dar a maior agilidade no projeto, pois este já foi finalizado, a ferramenta será de serventia pelo menos para os próximos, onde os usará como referência. Isto auxiliará as pesquisas, de forma a criar uma rede de conhecimento e, a partir de então, proporcionar a esperada maior agilidade na transformação dos protótipos de protocolos em partes fundamentais da Internet.

Para continuar a apresentação da solução é fundamental dizer que Codestand é o nome atual do projeto, porém inicialmente se chamava Codematch, mas teve seu nome trocado por conta de uma solicitação judicial acerca da patente do nome. Portanto, pode acontecer de eventualmente aparecerem termos como Codematch ou *Matches* em imagens ou *links* fornecidos nesta monografia.

3.0.1 A Estrutura dos usuários

Para melhor entender como funciona a dinâmica de interação, é necessário compreender a estrutura-base dos agentes no sistema. Deste modo, entende-se que a organização se dá em dois níveis de usuários:

1. Mentores - São membros mais experientes e que irão guiar o desenvolvimento da forma que acharem que deve ocorrer. É necessária a existência de permissão expressa no sistema para que o usuário seja um mentor.
2. Codificadores - São os entusiastas dos projetos e podem ser qualquer tipo de usuário com interesse em implementar. Estes irão codificar os projetos que acharem importantes, e que tenham sido descritos pelos mentores - ou seja, transformar em código as ideias abstratas relatadas.

Certamente que não existe impedimento para um mentor também ser codificador em outros projetos, mas é importante destacar que a recíproca não é necessariamente verdadeira. Por outro lado, não existe nenhuma restrição para que usuários que não sejam nem mentores e nem codificadores possam acessar a plataforma, de modo que usuários não registrados podem visualizar (não editar) todas as propostas e soluções disponíveis no Codestand. Seguindo na ideia de padrões abertos, quando alguém for beneficiado pelas informações dispostas, espera-se que exista real intenção de, no futuro, também contribuir - de forma que os ciclos se renovem por interesse da própria comunidade

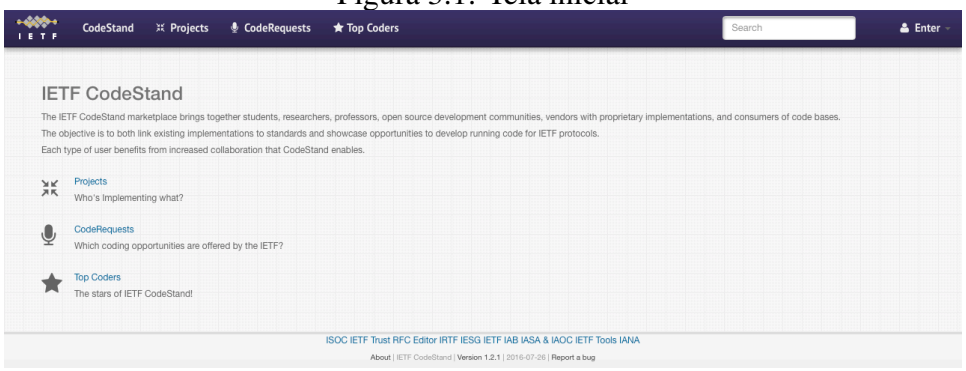
3.1 Como funciona

O Codestand aposta em uma interface simples, com uma divisão de telas que consiste basicamente no necessário para atender as funcionalidades dos mentores e dos codificadores. Uma parte adicional, provém ainda da teoria de gamificação, na tentativa de incentivar que os codificadores se sintam instigados a colaborar mais pelo ganho de reconhecimento dentro do Codestand. Logo, podemos dividir a solução em três partes essenciais: requisições de código, projetos e os melhores codificadores.

3.1.1 Uma primeira visão da interface

Antes de começar a apresentar de forma mais detalhada a essência do Codestand é preciso fazer uma familiarização do que há de básico na interface, para que se faça possível compreender os passos seguintes. Assim sendo, é necessário introduzir a tela inicial, o sistema de credenciamento e mais algumas características provenientes de ser uma aplicação incorporada ao Datatracker.

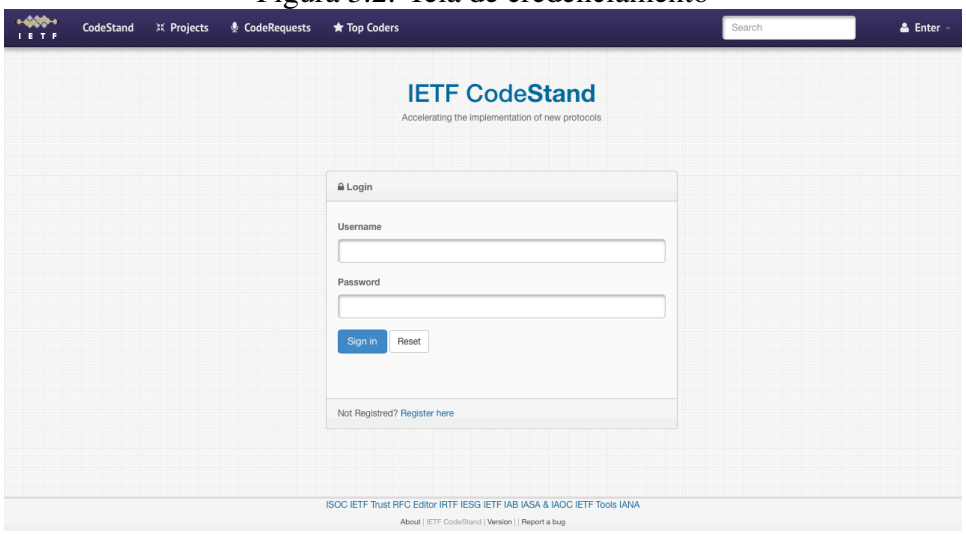
Figura 3.1: Tela inicial



Fonte: O Autor

Portanto, é possível ver através da Figura 3.1, a primeira tela que o navegador carrega na plataforma. Nela pode-se ver uma descrição sobre o que é o Codestand e *links* para as três funcionalidades principais, que serão descritas nas subsequentes seções (*Code Requests*, *Projects* e *Top Coders*).

Figura 3.2: Tela de credenciamento



Fonte: O Autor

Na Figura 3.2 é exibida a tela de login do Codestand que enviará para a validação no Datatracker.

Por questões de integração e maior facilidade de utilização, foi feita uma opção por um sistema de credenciamento reutilizado do Datatracker, de modo que, uma vez que se tenha conta no sistema do IETF, logo, o credenciamento será aplicado e funcionará igualmente no Codestand. Esta escolha proporciona uma facilidade de integração entre os sistemas, pois, é da intenção da plataforma que os usuários primeiro ajudem desenvolvendo códigos, para, em seguida sentir-se atraídos a escrever documentos para o IETF. Desta maneira, os usuários poderão manter apenas uma conta, o que se mostra muito mais prático.

3.1.2 Requisições de código

As requisições de código são conhecidas na plataforma como *Code Requests*, por isso, durante a presente monografia, ambos os termos serão utilizados de forma intercambiada. Os *Code Requests* correspondem às ideias de possíveis novos protocolos que são idealizados pelos mentores. Exemplificando, as requisições de código podem referir-se a um *Internet-Draft* para um novo protocolo, mesmo que ainda não se saiba realmente a efetividade da ideia. Assim, na medida que os codificadores implementarem as soluções, apontarem os problemas latentes e, através do refinamento oriundo da comunicação entre eles, consolidarem a proposta, é possível que esse *Internet-Draft* do protocolo evolua em uma RFC e seja em um futuro breve um novo padrão na área de redes.

Sendo o *Code Request* uma tarefa exclusiva dos mentores, e sabendo que o objetivo é atrair os desenvolvedores ao seu projeto, o mesmo deve ser extremamente didático, e completo, ao realizar o cadastro na ferramenta. Isto porque sua tarefa é exatamente instruir aos codificadores como e porque o novo protocolo deve ser implementado. A política de permissões ainda não está bem definida, mas já existe na plataforma a possibilidade de restrição para tal. É provável que, em breve, esta política já vigore corretamente, porém, atualmente, qualquer pessoa está autorizada a ser um mentor.

Uma vez como mentor, o mesmo pode solicitar um *Code Request*, que é requerido através de um formulário solicitado na tela de requisições de código,

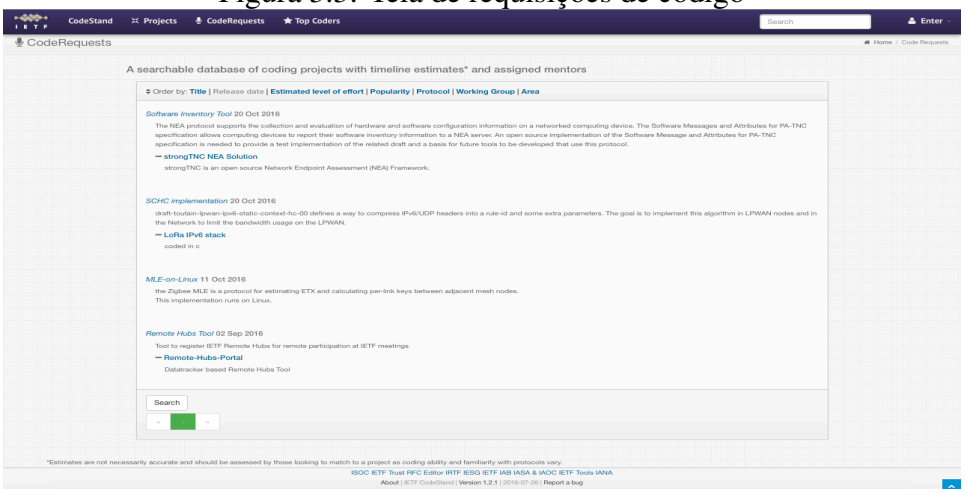
Alguns pontos são indispensáveis na solicitação via formulário, como:

- Título e Descrição - Informam o nome e como de fato se espera que o protocolo seja implementado, respectivamente.
- Protocolo - Apesar de a plataforma ser destinada a gerenciar novos protocolos, o

título não necessariamente diz respeito ao protocolo de rede que está sendo desenvolvido. Isto ocorre para gerar maior número de opções entre os protocolos apresentados. Um exemplo: vários usuários poderiam, teoricamente, propor um IPv8, de formas diferentes.

- Mentor - Todo projeto necessita, obrigatoriamente, de um mentor, mas este não precisa ser exatamente quem o requisita. É importante que seja um mentor a fazer a solicitação, mas o mesmo pode delegar a outra pessoa (com permissão) a tarefa de acompanhar o andamento do projeto.
- *Internet-Drafts/RFCs* - Indica quais documentos do IETF estão relacionados a este projeto. Deve seguir a ideia do *Rough Consensus,* e, imaginando que ele deve acontecer em paralelo, ao *Running Code* - o que significa ter alguém trabalhando no documento formal de um lado, enquanto no Codestand já há usuários trabalhando em sua implementação. Além disso, deve ser possível correlacionar outros documentos já consolidados como base teórica para este novo projeto.
- Nível de esforço - Indica mais ou menos quanto tempo o Mentor espera que se gaste para implementar o projeto. Sendo assim, os codificadores podem estimar se vale a pena ou não pegar o projeto, baseado no tamanho do período que deverão empregar na tarefa.
- *Tags* - Como a aplicação tem um viés de rede social, a ideia é permitir buscas através de palavras chaves definidas em *tags*.

Figura 3.3: Tela de requisições de código



Fonte: O Autor

A Figura 3.3 ilustra a aplicação atual do Codestand, de modo que já está em utilização real, e por isso é possível ver diversos projetos já representados na lista. Na imagem referida, é possível visualizar o título, a descrição e a data de criação dos projetos, e em al-

guns deles já existem inclusive projetos relacionados, mas que serão abordados com mais detalhes na seção seguinte.

Porém, neste momento, o importante é destacar que, através dessa tela, os codificadores devem procurar pelos projetos que mais lhe atraem e então se associarem a estes. Ainda olhando para a Figura 3.3, é perceptível o fato que, por ser simplesmente uma lista de projetos, existem diversos tipos de ordenação possível para que o codificador possa procurar especificamente pelo que lhe interessa. Dentre eles, se destacam aspectos como: Popularidade, Grupo de trabalho e Área.

O primeiro enumera os projetos de cada *Code Request*, avaliando os que possuem mais projetos associados aos que possuem menos, de modo que os codificadores podem confiar que aquele *Code Request* é sobre um tópico bem requisitado e provavelmente está bem documentado. O segundo e o terceiro, Grupo de Trabalho e Área, são ocasionados devido aos *Internet-Drafts*/RFCs que foram ligados ao projeto. Como todo documento do IETF possui um grupo de trabalho e uma área, logo, a requisição de código pertencerá de certa forma a todos os grupos de trabalhos e áreas correlacionados aos seus documentos. Isto, por si só, acaba se tornando um bom filtro para o codificador - se, por exemplo, ele for um entusiasta da área de segurança, o mesmo pode agrupar os *Code Request* pela área desejada e então refinar a busca por um projeto dentro apenas da referida área.

Quando se é um mentor, pode se aplicar ainda dois tipos de filtros sobre a lista. O primeiro, para filtrar as requisições de código pelas que o próprio mentor requisitou, e assim, é possível acompanhar o andamento mais próximo de todos no qual ele mesmo descreveu através do formulário. E o segundo concerne as requisições de código em que o mesmo é o mentor - lembrando que requisitar e ser o mentor não necessariamente implicam na mesma coisa, pois é possível requisitar o projeto, mas designar outro mentor para a tarefa de supervisionar.

3.1.3 Projetos

A parte central da plataforma - e que representa de fato o que é buscado pelo IETF - são os projetos. Assim como para os *Code Requests*, pode ser que os projetos sejam chamados de *Projects*, pois é a nomenclatura adotada dentro da plataforma. Os projetos são a materialização das requisições de código em implementações, que servirão como base para avaliar se a proposta do novo protocolo é realmente relevante ou não. Isto significa que o *Running Code* que guia o IETF está representado através desta funcionalidade - ou

seja, é neste ponto que se deve pôr à prova os documentos propostos.

Algo que precisa ser destacado aqui, é que os projetos podem tanto representar implementações das requisições de código em diversas linguagens, como servir como documentação de protocolos já consolidados no passado. Este segundo tipo é conhecido como *Stand Alone*, no qual não existe uma correlação com nenhum *Code Request*, visto que já está fechado e funcionando. Este caso serve mais para dar referências aos novos projetos que estão por vir. Mas, em geral na plataforma, o que se quer é que os projetos venham de pessoas que leram as descrições das requisições de código e que decidam, por isso, implementar e colocá-los à prova.

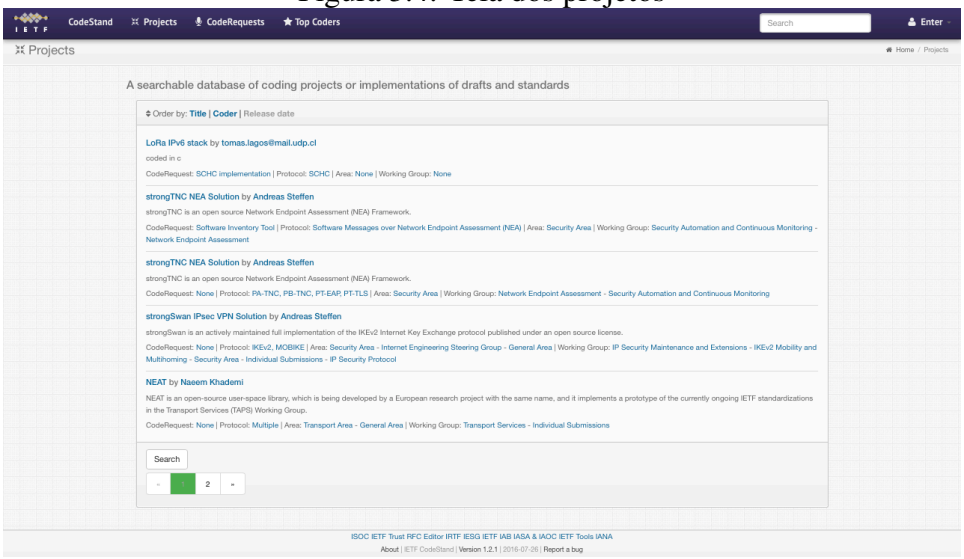
Uma vez que o codificador encontrou uma requisição de código e decidiu associar se aquele projeto, a partir daí ele precisa preencher um formulário com algumas informações básicas:

- Título e descrição - Indicam o nome do seu projeto e a forma como pretende solucionar o problema proposto, respectivamente.
- URL - O Codestand, como já foi dito anteriormente, não é uma ferramenta para que as implementações sejam feitas dentro dele. Portanto URL aponta para o repositório onde será mantido o andamento do projeto - como um Github (versionador de código), por exemplo.
- *Tags* - Assim como nos *Code Requests*, as *tags* servem para facilitar a busca dos usuários que irão procurar através dos projetos por soluções para seus problemas dentro e fora da plataforma.

Quando cadastrado, um projeto *Stand Alone* funciona como se estivesse cadastrando no mesmo formulário as informações referentes à requisição de código (vistos na seção anterior), mais as informações do projeto. Portanto, o mesmo será um projeto, mas conterá informações de *Code Request* como *Internet-Drafts* e RFCs relacionadas, e, por conseguinte, áreas e grupos de trabalho. Porém, não possuirá mentor e nem nível de esforço, visto que está sendo cadastrado apenas a título de documentação.

Como pode ser visto na Figura 3.4, a tela de projetos é muito semelhante a de requisições de código. Basicamente, consiste de uma lista com os projetos relacionados, onde é possível ver que, abaixo de cada projeto, são enumeradas algumas informações importantes como o protocolo correspondente, os grupos de trabalhos e as áreas associadas aos documentos do *Code Request*. É possível perceber que alguns constam com *Code Request* igual a *None* (nenhum), pois estes são os casos de projetos *Stand Alone*.

Figura 3.4: Tela dos projetos



Fonte: O Autor

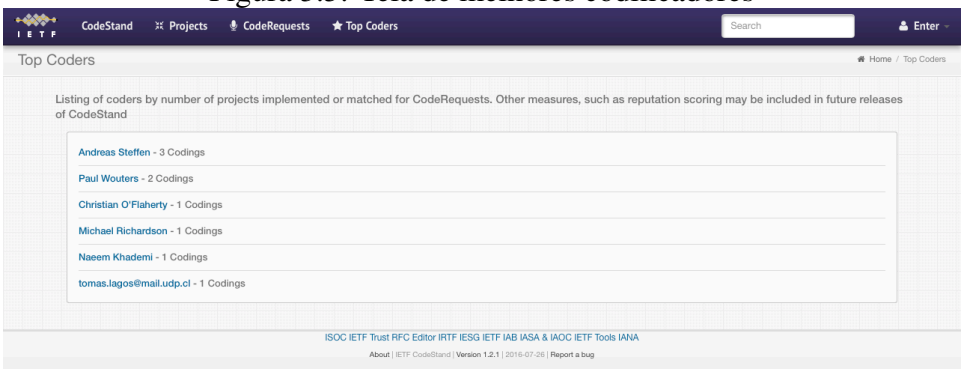
3.1.4 Melhores codificadores

Existe ainda uma terceira parte do Codestand que não se refere nem a requisições de código e nem a implementações destas. No sistema, ela é conhecida como *Top Coders* (principais codificadores), e é uma primeira tentativa de utilizar gamificação para atrair mais codificadores ao Codestand.

Apesar de ainda não existir nenhuma grande recompensa ao usuário por este ter cadastrados mais projetos que os outros, nesta parte da plataforma é possível ver uma lista com os maiores codificadores. Como em diversos games, o objetivo aqui é tentar estimular a ideia de ranking (posição), ou seja, instigar os codificadores a implementarem mais projetos para que seu nome esteja em destaque. Considerando que o Codestand é uma ferramenta para o IETF, ter seu nome em destaque na plataforma pode significar um ganho de visibilidade. Por exemplo, um aluno de graduação que ainda não sabe como funciona o *Internet Engineering Task Force*, mas que tem interesse, pode usar da ferramenta como plataforma para tornar seu nome mais forte e por conseguinte facilitar o *networking* com outros membros. O objetivo é que, no futuro, outras formas de estímulo sejam propostas, como uma pontuação pela *reputação* de seus projetos, ou seja, uma maneira de, de alguma forma, avaliar se a implementação proposta foi realmente bem elaborada.

Como pode ser visto na Figura 3.5, existe uma lista na qual a ordem definida vai dos usuários que mais codificaram para os que menos codificaram. No momento, como temos poucos projetos cadastrados na plataforma, é possível ver todos usuários, mas a ideia é que, com o passar do tempo, constem nesta lista apenas os principais usuários.

Figura 3.5: Tela de melhores codificadores



Fonte: O Autor

3.2 Casos de Uso

Agora que já se conhece as funcionalidades da plataforma e foi apresentada uma ideia do fluxo entre as telas, serão apresentados, nesta seção, dois exemplos que retratam possíveis cenários de uso do Codestand. Desta forma, é mais fácil de se obter uma noção completa do desenvolvimento do processo, desde a ideia do mentor, até o cadastro e, pôr fim, no momento da implementação.

3.2.1 Exemplo 1 - Um caso acadêmico

Para explicar melhor em que contexto poderia se suceder a utilização do Codestand, é interessante relatar um caso que poderia muito bem acontecer dentro de uma universidade. Mesmo sabendo que é uma plataforma direcionada especificamente ao IETF e ao seu desenvolvimento de padrões, o Codestand pode ser usado por professores que lecionam disciplinas de redes se, por ventura, desejarem dar um viés mais prático a algum trabalho de aula.

Por exemplo, digamos que o trabalho da disciplina de Redes deste semestre fosse desenvolver um protocolo chamado TCP2 e que este fosse alguma pequena modificação no protocolo TCP (DARPA; USC, 1981) já existente. Neste cenário, o professor ficaria encarregado de ser o Mentor da turma, logo, ele iria entrar na tela de requisição de código e cadastrar uma nova proposta. Esta requisição de código deveria conter um título do trabalho e a descrição elaborada do trabalho da disciplina, informando detalhadamente o que é esperado. Esta requisição de código possivelmente estaria ligada a RFC correspondente ao protocolo TCP (RFC 793) e o tempo esperado de esforço seria o tempo de duração do trabalho da disciplina.

Portanto, os alunos, para realizarem o trabalho, deveriam ir na tela de *Code Requests*, fazer a busca pelo trabalho e associarem seus projetos. Eles deverão cadastrar o título de seu trabalho, a forma como pretendem fazer e o link para o site de versionamento de código escolhido, pelo qual o professor irá acompanhar o andamento do trabalho. Desta forma, o professor poderia, através da página de requisições de código que ele monitora, entrar no seu *Code Request* e conferir cada um dos projetos associados. Ao final da disciplina, a avaliação seria feita por fora do sistema para aprovar os alunos, mas o conhecimento gerado ficaria na plataforma para que turmas futuras, e quem mais tivesse interesse, também pudesse se beneficiar.

3.2.2 Exemplo 2 - Dentro do IETF

Foi escolhido como primeiro exemplo um caso acadêmico, pois até que o Codestand se consolide completamente, é o mais próximo do que deve acontecer. Na prática, algumas mudanças devem ocorrer, e elas serão retratadas neste segundo caso de uso.

No momento, a tendência é que o professor seja já membro do IETF e que esteja escrevendo algum documento para a organização, seja um *Internet-Draft* ou mesmo uma RFC. Pois bem, a dinâmica será parecida com o caso anterior, mas aqui os alunos escreverão seus trabalhos por serem bolsistas do tal professor, tanto por iniciação científica, mestrado ou doutorado. Portanto, neste exemplo, o resultado final será não a aprovação dos alunos, mas sim realmente auxiliar o professor na sua tarefa de provar o *Running Code* e, mais que isso, mostrar que a sua ideia de novo protocolo é realmente pertinente. Porém, mesmo tendo algumas pequenas diferenças nos objetivos dos professores, a forma como se dará a interação com a plataforma, tanto do professor quanto dos alunos, é exatamente a mesma. Isto permite uma flexibilidade para o Codestand, à medida que, em diversos cenários, o fluxo de trabalho é o mesmo. E, também neste caso, o resultado ficará armazenado na plataforma e poderá ser usado por todos.

4 A IMPLEMENTAÇÃO

Por ser uma plataforma Web, o Codestand se utilizou de técnicas de desenvolvimento de software, além de *frameworks*, bibliotecas e tecnologias que permitiram que se chegasse ao estado atual do projeto. Portanto, neste capítulo serão abordados todos os aspectos que constituem a identidade da plataforma, como sua arquitetura, as tecnologias que dispõe, qual processo de desenvolvimento utilizado, entre outras coisas que se referem especificamente a implementação do ponto de vista de código e a base de dados do Codestand.

O Codestand é uma aplicação incorporada ao Datatracker, permitindo que muitas das suas tecnologias, além de código e base de dados sejam reutilizadas do Datatracker. Conceitualmente é uma aplicação totalmente diferente, com interface própria e suas próprias características. Ele se aproveita, no entanto, do fato de ser incorporada a esta outra aplicação para reutilizar alguns controles e tecnologias já desenvolvidas, além do sistema de credenciamento e, claro, reutilizar no Codestand dados cadastrados no Datatracker, como documentos e usuários.

4.1 As Tecnologias

A linguagem de programação escolhida para a implementação da plataforma foi Python 2.7 (PYTHON27, 2016), mais especificamente Django que é um *Framework Web* para Python de alto nível, e que estimula o rápido desenvolvimento, o código limpo e um design pragmático (DJANGO, 2016). A referida linguagem foi escolhida pois possui uma documentação ampla e uma comunidade bastante ativa, o que facilita para os desenvolvedores, mesmo que estes não possuam nenhum conhecimento específico em Python ou Django.

Por ser uma aplicação Web, foi necessário o uso de um banco de dados integrado à plataforma e para isso, o escolhido foi o MySQL (MYSQL, 2016), principalmente por ser o banco de dados de código livre mais utilizado no mundo e portanto, extremamente bem consolidado e documentado. Para fazer a comunicação entre o código Django e a base de dados MySQL foram usadas algumas bibliotecas específicas para fazer esta integração, assim como para realizar a parte da comunicação entre *front end* e *back end*. Por questão de simplificação a maioria destas bibliotecas de intermediação serão ocultadas desta monografia.

O principal para conseguir instalar e desenvolver o Codestand é possuir Django e MySQL, o que pode ser feito gratuitamente por qualquer pessoa, visto que são plataformas de código aberto. Se for do interesse do leitor participar do Codestand, ou mesmo conhecer mais a fundo a tecnologia, o código foi hospedado na plataforma de versionamento Github e pode ser encontrado em (CODEMATCH-GITHUB, 2016). O Github (GITHUB, 2016) abriga uma das maiores coleções de código aberto e também por isso foi a escolhida para hospedar a plataforma descrita nesta monografia.

Uma vez que se sabe de onde baixar, quais as principais tecnologias utilizadas e um pouco do necessário para desenvolver o Codestand, ou seja, os fatores de implementação foram minimamente explicados, é importante também conhecer sobre os aspectos referentes a implantação da plataforma no ambiente de produção atual. Apesar de ser possível instalar o Codestand em qualquer ambiente, é mais simples o fazê-lo em servidores UNIX, pois as tecnologias necessárias, assim como algumas bibliotecas anteriormente suprimidas são mais facilmente obtidas nestes sistemas operacionais. Para hospedar a aplicação foi necessário utilizar um servidor HTTP. O servidor HTTP escolhido foi o mais famoso Web Server do mundo, o Apache Server (APACHE, 2016). O Apache pode ser instalado em ambiente Windows, mas mais uma vez é importante destacar que possivelmente é mais simples instalá-lo e rodá-lo em ambientes UNIX. Portanto, atualmente o Codestand está rodando em um servidor Linux da Amazon, com Apache sendo o servidor Web, Django o framework para rodar o código, MySQL a base de dados da plataforma e Github o sistema de versionamento para permitir que as mudanças feitas em desenvolvimento entrem em vigor em produção.

4.2 Processo de desenvolvimento

Um dos fatores determinantes para o bom andamento do projeto é o fato dele possuir metas bem traçadas desde o início, de modo que as iterações de desenvolvimento puderam ser particionadas. Pode se dizer que a técnica de desenvolvimento utilizada foi baseada em Scrum (SCRUM, 2016), no qual um Product Owner - no caso eram alguns membros da ISOC e do IETF que faziam este papel - definiram as atividades e as prioridades em três etapas. Na primeira etapa constam as funcionalidades básicas da plataforma, as que podem ser consideradas o requisito mínimo, ou seja, as quais são essenciais para a utilização do Codestand. Como exemplo desta etapa, podemos destacar o cadastro de requisições de código e a possibilidade de associar projetos às tais requisições.

A segunda etapa prevê tarefas de melhoria das funcionalidades definidas na etapa anterior, de modo que facilitem a vida do usuário e que tornem o Codestand mais atrativo. Além disso, estão previstas algumas funcionalidades estatísticas para que os administradores tenham dados para melhorar ainda mais a plataforma. Nesta etapa se destacam a geração de gráficos a partir do uso da plataforma e a possibilidade de ligar as *tags* dos projetos e das requisições de códigos diretamente aos eventos promovidos pelo IETF. Para terceira, e, por enquanto, última etapa estão as funcionalidades avançadas, que são aquelas que seriam interessantes se existissem, mas que não são indispensáveis aos usuários. Por exemplo, a possibilidade de os codificadores receberem notificações de erratas nos *Internet-Drafts* ou RFCs que estão ligadas ao seu projeto.

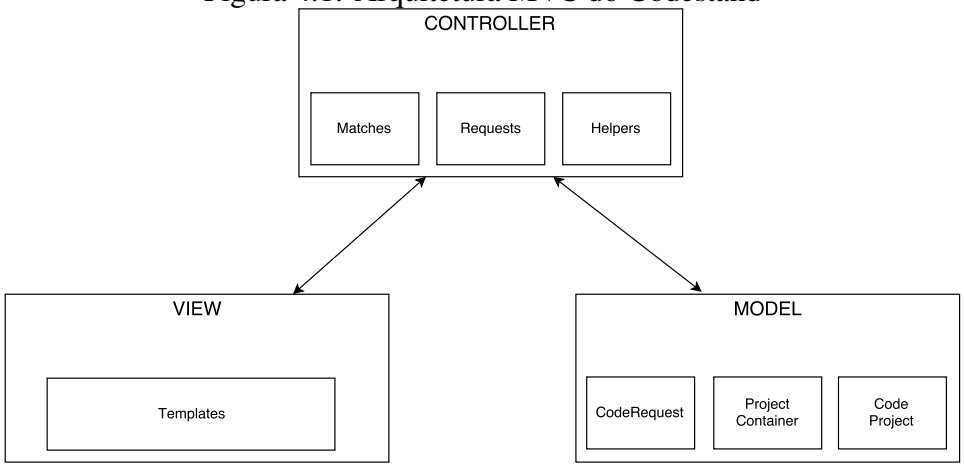
Como metodologia de desenvolvimento, as atividades foram fragmentadas em partes menores e agrupadas em *sprints* de uma semana. Assim, a cada quarta-feira, acontece uma reunião para apresentar os resultados e planejar a próxima semana, corrigindo o que não deu certo e seguindo o planejamento adiante. Para que o desenvolvimento pudesse seguir, mas, ao mesmo tempo, estivesse sempre disponível para alguma eventual apresentação, foram tomadas algumas medidas. A principal delas foi criar, dentro do servidor, dois ambientes de produção, sendo o primeiro de desenvolvimento. Ou seja, para as reuniões semanais sempre seria possível acessar este para ver o que foi feito no decorrer da semana. No outro, se manteve sempre versões mais estáveis, portanto, que tinham sido testadas e recebido um aval dos gerentes do projeto. Assim, caso fosse necessário apresentar em algum evento, certamente seria utilizada a versão citada neste segundo caso. E mais, os testes realizados sobre um ambiente considerado estável permitem maior confiança já que os problemas encontrados não são decorrência de algo transitório, criado por razão do desenvolvimento.

Essa separação em dois ambientes de produção foi facilitada pelo Github, que permite a criação de ramificações. Com isso, o codificador não precisa realizar manualmente a transição da versão de desenvolvimento para a considerada estável.

4.3 Sobre o Código

O *Framework Web* Django que foi utilizado para desenvolver o código do Codestand utiliza a arquitetura MVC e até por isso foi optado por desenvolver orientado a este paradigma. De modo que a Figura 4.1 representa a arquitetura MVC que foi aplicada ao Codestand.

Figura 4.1: Arquitetura MVC do Codestand



Fonte: O Autor

4.3.1 Controlador

Como pode ser visto na Figura 4.1, o *Controller* (Controlador) é o orquestrador, ou seja, ele faz a comunicação entre a *View* (Visualização) e o *Model* (Modelo). O controlador exerce a função de código *back end*, ou seja, é o responsável por captar os dados do Modelo e renderizá-los juntos aos *templates* da Visualização. No Codestand o controlador está representado através de três grandes classes:

- *Matches* - Dizem respeito ao código dos projetos. Neste trecho de código que é gerenciada a inserção e edição dos projetos, tanto quando o projeto é *Stand Alone*, quanto quando ele está sendo associado via uma requisição de código previamente cadastrada.
- *Requests* - Representam as requisições de código, portanto quando um usuário decidir fazer o cadastro de uma nova solicitação é esta classe que deve gerenciar a inserção e edição de tal proposta.
- *Helpers* - São os códigos mais genéricos, a maior parte é constituída de código em comum entre *Matches* e *Requests*. Também possui código que não diz respeito a alguma funcionalidade específica, como renderizar a tela inicial, chamar as funções de login via Datatracker e exibir a tela de *Top Coders*.

A Figura 4.2 representa todas as funções relacionadas aos *Matches*, e por conseguinte, tudo que é possível fazer através do Codestand no que diz respeito aos projetos. Portanto, *show list* (exibir lista) mostra a lista de todos os projetos cadastrados, sendo os argumentos algumas variáveis possíveis, como a ordenação utilizada na lista, se a lista deve exibir apenas os projetos cujo o usuário é o autor e qual a página está sendo exibida.

Figura 4.2: Código dos *Matches*

```
def show_list(request, is_my_list="False", att=constants.ATT_CREATION_DATE, state="False", page=1):

def show(request, pk, ck):

def search(request, is_my_list="False"):

def save_code(request, template, pk, ck="", coding=None):

@login_required(login_url=settings.CODESTAND_PREFIX + constants.TEMPLATE_LOGIN)
def edit(request, pk, ck):

@login_required(login_url=settings.CODESTAND_PREFIX + constants.TEMPLATE_LOGIN)
def new(request, pk=""):

@login_required(login_url=settings.CODESTAND_PREFIX + constants.TEMPLATE_LOGIN)
def remove_link(request, ck, link_name):

@login_required(login_url=settings.CODESTAND_PREFIX + constants.TEMPLATE_LOGIN)
def remove_tag(request, ck, tag_name):

@login_required(login_url=settings.CODESTAND_PREFIX + constants.TEMPLATE_LOGIN)
def remove_document(request, pk, doc_name):
```

Fonte: O Autor

As funções *new* (novo) e *edit* (editar) representam o cadastro de um novo projeto e a edição de algum projeto, respectivamente. A chamada de *save code* (salvar código) realiza a transferência do projeto para a base de dados, seja ele um novo ou apenas uma atualização de algum já existe. É referido como salvar código pois conceitualmente cada projeto representa uma implementação. A chamada de *search* (buscar) permite filtrar os projetos que são exibidos de acordo com alguma característica que interessem ao usuário, como por exemplo buscar os projetos codificados por alguém em específico. A função *show* (mostrar) exibe os dados relativo a algum projeto cadastrado, ou seja, permite visualizar com mais detalhes o que foi descrito naquele projeto.

As funções com prefixo *remove* são apenas funções auxiliares para remover alguns dados durante a operação do formulário em inserção/edição. Além disso, é possível ver, em cima de algumas funcionalidades, a diretiva *login required* (credenciamento requerido) que indica que esta função só pode ser chamada por usuários logados, ou seja, se alguém tentar forçar a chamada via navegador será direcionado para a tela de credenciamento.

A ideia durante a implementação do Codestand foi simplificar ao máximo as funcionalidades, tendo o menor número de funções. E mais que isso, houve um grande esforço para que os projetos e requisições de código fossem o mais semelhante possível. Portanto, é possível ver através da Figura 4.3 a semelhança com a assinatura das chamadas de funções da Figura 3.4.

Para não repetir minuciosamente o que faz cada função, é plausível afirmar que os *Code Requests* tem as funções análogas aos *Matches*, ou seja, permite exibir, inserir, editar, procurar e visualizar as requisições de código. É possível perceber, entretanto, que as funções que começam pelo prefixo remove são diferentes entre as Figuras 4.2 e 4.3, pois

Figura 4.3: Código dos *Requests*

```
def show_list(request, type_list="all", att=constants.ATT_CREATION_DATE, state="False", page=1):

def search(request, type_list="all"):

def show(request, pk):

def save_project(request, template, project_container=None):

@login_required(login_url=settings.CODESTAND_PREFIX + constants.TEMPLATE_LOGIN)
def edit(request, pk):

@login_required(login_url=settings.CODESTAND_PREFIX + constants.TEMPLATE_LOGIN)
def new(request):

@login_required(login_url=settings.CODESTAND_PREFIX + constants.TEMPLATE_LOGIN)
def remove_contact(request, pk, contact_name):

@login_required(login_url=settings.CODESTAND_PREFIX + constants.TEMPLATE_LOGIN)
def remove_document(request, pk, doc_name):

@login_required(login_url=settings.CODESTAND_PREFIX + constants.TEMPLATE_LOGIN)
def remove_tag(request, pk, tag_name):
```

Fonte: O Autor

as características de projetos e requisições de código são diferentes durante a operação do formulário de inserção/edição.

O código referente ao *Helpers* não será explicitamente citado, pois em geral são funções auxiliares e as grandes estrelas do Codestand são os Projetos representados pelos *Matches* e as Requisições de código representadas pelos *Code Requests*.

4.3.2 Visualização

Conforme dito anteriormente, o controlador exerce as funções de capturar os dados provenientes do modelo, processá-los e, em seguida, renderizar os templates da visualização, conforme pode ser visto na Figura 4.1.

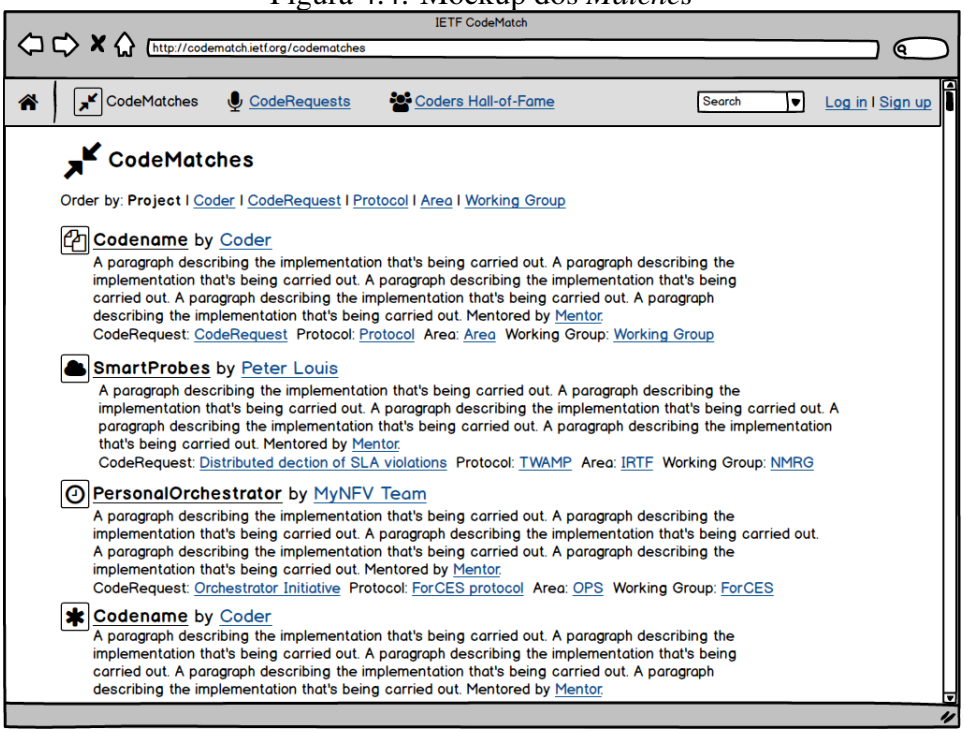
Estes *templates* representam o código HTML entrelaçado com o código Django, da mesma forma como a maioria das linguagem orientadas a Web fazem, como por exemplo o PHP. De modo que, quando o controlador deseja apresentar alguma interface na tela, ele chama uma função de renderização e passa como argumento algumas variáveis de contexto para permitir que os dados captados do modelo sejam apresentados na visualização HTML.

O Codestand possui uma interface de poucas telas e que já foi apresentado na sua maioria no capítulo anterior ao abordar o funcionamento da plataforma. Portanto, as telas apresentadas nessa seção tem como objetivo estabelecer um paralelo entre como estão atualmente e como foram pensadas inicialmente, de modo a apresentar também a dinâmica de evolução da parte gráfica.

No andamento da interface gráfica do Codestand, desde o início até o presente

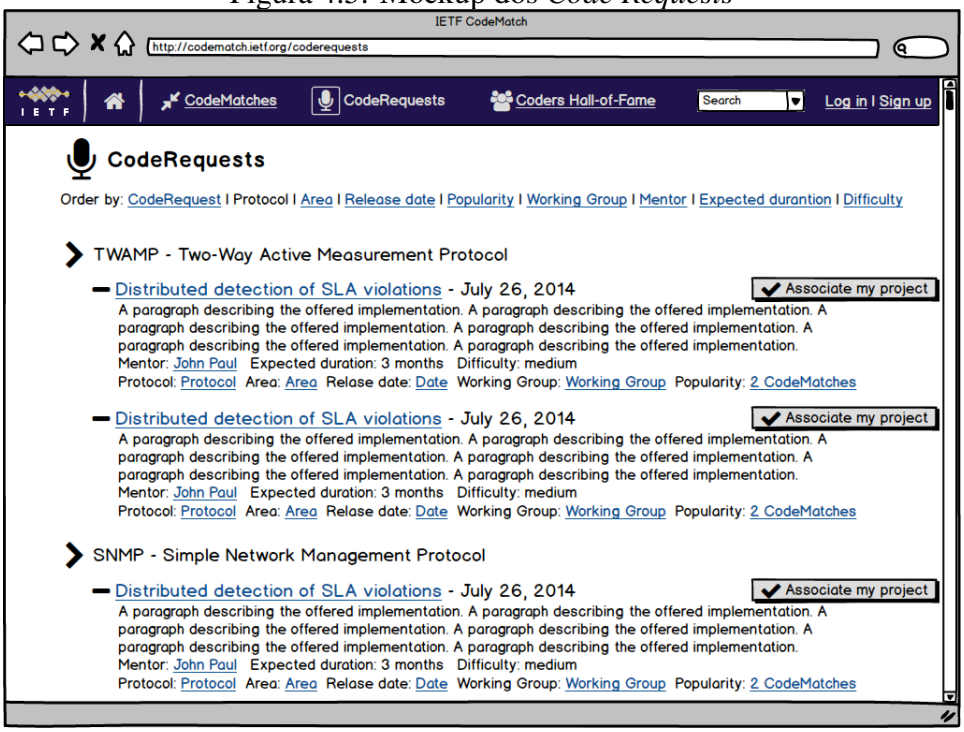
momento, destaca-se principalmente o uso de *mockups* que permitem que as telas sejam esboçadas sem escrever uma linha em HTML ou qualquer outra linguagem interpretável pelo navegador.

Figura 4.4: Mockup dos *Matches*



Fonte: Grupo de trabalho do IETF

Figura 4.5: Mockup dos *Code Requests*



Fonte: Grupo de trabalho do IETF

As Figuras 4.4 e 4.5 são os *mockups* desenvolvidos pelo grupo de trabalho responsável pelo Codestand no IETF. Estes *mockups* representam basicamente o que existe hoje na interface, salvo algumas diferenças como as nomenclaturas, por exemplo *Coders of Hall-of-Fame* que mudou para *Top Coders* e *CodeMatches* que virou *Projects*. Porém, as semelhanças são bastantes, o que permite inferir que pelo menos em termos gráficos, já se tinha bem consolidado o que se queria ao início do projeto.

Ao transformar os *mockups* em código para serem incorporados ao Codestand foi utilizada uma das principais tecnologias conhecidas hoje que é o Bootstrap (BOOTSTRAP, 2016), que é o mais popular *framework* de HTML, CSS e Javascript na Web. De forma que também a implementação da interface gráfica do Codestand é feita utilizando tecnologia de ponta.

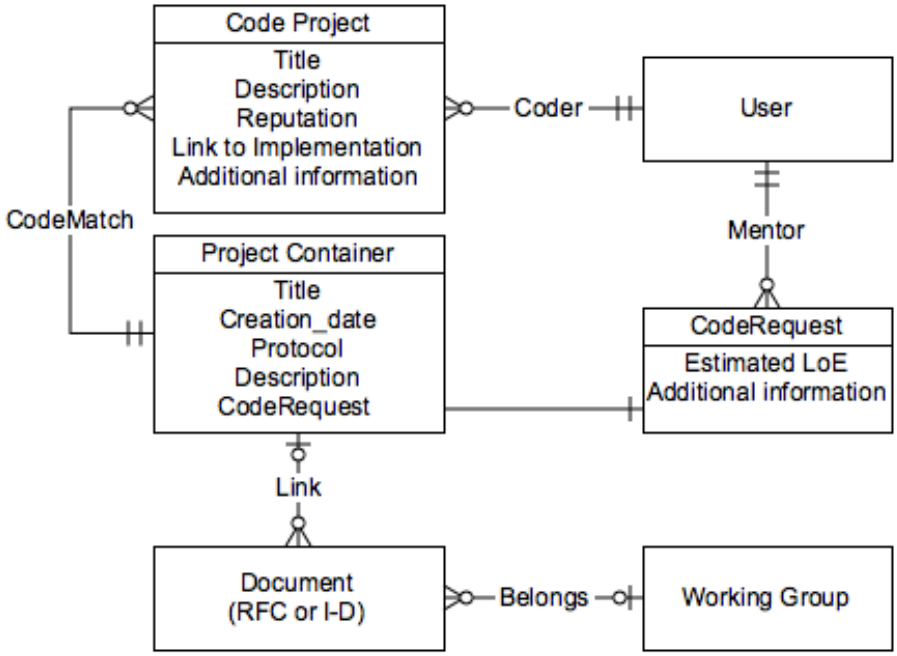
Existe hoje uma grande preocupação da ISOC e do IETF no que se refere à interface, alegando que esta deve ser o grande diferencial em captar e resolver os problemas que o Datatracker não consegue. Apesar de ser possível estabelecer algum paralelo entre a interface do Codestand e a do Datatracker, o objetivo aqui é criar um ambiente diferente, com o estilo de uma rede social e que permita que as ações sejam realizadas facilmente.

Sabendo da necessidade de cativar o público, o Codestand tem frequentemente solicitado que alguns usuários não familiarizados com a plataforma a utilizem e, então, dêem sua opinião sobre a ferramenta. Estes experimentos, os resultados e o futuro imaginado da interface serão retratados no capítulo 5.

4.4 A Base de Dados

A base de dados também faz parte do esquema MVC, pois é a parte da arquitetura que trata dos modelos de dados que serão armazenados na plataforma, e que posteriormente serão exibidos na interface gráfica. Mas, por possuir mais nuances que o Controlador e a Visualização, será tratado em uma seção a parte, demonstrando sua arquitetura, a interação com a base de dados do Datatracker e a forma como Django permite utilizar-se do banco de dados - que é, provavelmente, a principal técnica de armazenamento de dados.

Figura 4.6: Código dos *Matches*



Fonte: Grupo de trabalho do IETF

4.4.1 Os modelos do Codestand

A Figura 4.6 apresenta a modelagem conceitual da base de dados do Codestand, de modo que é possível perceber três principais estruturas de dados que serão armazenadas de acordo com as entradas do usuário na plataforma.

Supondo que o usuário seja um mentor, logo, ele entrará na plataforma e cadastrará um tipo de dados conhecido como *CodeRequest*, ou seja, uma requisição de código para a ideia que ele teve. Assim sendo, esta estrutura *CodeRequest* contém os dados referentes ao que o mentor imagina de dificuldades do projeto, como por exemplo o tempo estimado. Ao criar um *CodeRequest* ele também preenche as informações relativas ao *Project Container*, sendo possível perceber pela associação da Figura 4.6 que um *CodeRequest* sempre estará associado a um *Project Container*. Desta forma, o *Project Container* é que mantém as informações relativas ao título, descrição e protocolo referentes à requisição de código. Além disso, o *Project Container* possui relação com documentos do IETF, sejam eles *Internet-Drafts* ou RFCs, e todo documento é, por sua vez, associado a um grupo de trabalho. Logo, é possível associar diretamente um *CodeRequest* aos grupos de trabalho no qual este está correlacionado.

Caso o usuário seja um codificador, ele criará na plataforma um *Code Project*, e assim, ele vai associar sua implementação a um *Project Container*, sendo este o processo algumas vezes já descrito de associar um projeto à um requisição de código. A estrutura de dados *Code Project* possui, portanto, as informações relativas a implementação que o usuário está realizando, como título, descrição e a URL que o liga ao site de versionamento.

4.4.2 A configuração da base de dados

O Django possui alguns mecanismos interessantes de conexão com base de dados que merecem ser destacados. O principal deles é o arquivo de configuração *settings.py*, no qual são descritas as características da plataforma, como variáveis de ambiente e, principalmente, as informações de base de dados.

Neste arquivo de configuração, é permitido escrever as definições de conexão com a base de dados e, mais que isso, é possível setar para a aplicação utilizar múltiplas base de dados. No caso do Codestand, esta opção é fundamental, uma vez que parte da base de dados é compartilhada com o Datatracker, que, pelo motivo de ainda estar em desenvolvimento, não pode fazer a implantação no servidor onde roda o Datatracker.

O *Framework Web* Django gerencia as consultas feitas pelo banco de dados, não sendo necessário escrever uma única linha de código SQL. Todas as consultas são realizadas através da API do Django, e este mesmo entende quando a tabela pertence ao Codestand ou quando ela pertence à aplicação externa, que, no caso, é o Datatracker. Sendo as consultas SQL feitas através da API do Django, diversos problemas de seguranças deixam de ser preocupação do codificador da plataforma. Esta proteção minimiza significativamente a chance de ataques mais básicos à plataforma, como *SQL Injection* (SQL-INJECTION, 2016).

4.4.3 Integração com Datatracker

Para que o Codestand seja uma aplicação poderosa ao IETF, ela necessita, obrigatoriamente, acessar dados do Datatracker e, mais que isso, ela precisa que estes dados estejam atualizados, de modo que o credenciamento no Datatracker seja aceito no Codestand. Além do fato de que os documentos ligados pelos mentores precisam ser os mais atuais possíveis, para que o codificador não tenha dúvidas na hora de escolher o projeto,

evitando trabalhar em algo cujos erros já foram sanados nas versões mais recentes do documento.

Quando um usuário acessar o Datatracker para se cadastrar, obrigatoriamente as credenciais que ele criar precisam estar, automaticamente, válidas após o processo. Deste modo, o Codestand precisa que parte da sua base de dados esteja dispostas no Datatracker, ou seja, que algumas informações venham da aplicação do IETF. Portanto, as tabelas que representam os usuários e os documentos, respectivamente, estão localizadas em outro local físico da Internet, o que possivelmente gere atrasos durante a utilização - e que será tópico de discussão na seção 5.

Em especial sobre o credenciamento, pelo fato de estar todo disposto em outra aplicação, isto também reduz potencialmente os problemas de segurança do Codestand, uma vez que o Datatracker é uma aplicação muito mais consolidada e que conta com diversos especialistas de segurança envolvidos.

5 AVALIAÇÃO

Atualmente, após pouco mais de um ano de projeto, o Codestand se encontra em um estágio de implementação que pode ser considerada avançado - visto que em Julho de 2015 (início do projeto) foi estabelecida uma lista de implementações que deveriam ser seguidas até que o Codestand pudesse ser considerado em versão funcional.

O objetivo era que esta primeira versão funcional pudesse ser considerada finalizada e, por conseguinte, apresentada na reunião do IETF-96 (IETF-96, 2016) em Berlim, em Julho de 2016. A motivação foi apresentá-lo em um dos eventos paralelos da reunião, onde são tomadas novas decisões sobre os padrões e introduzidos novos protocolos. Considerando, então, que o primeiro objetivo do Codestand foi concluído, com sucesso, conforme o planejamento, é possível olhar o estado atual e dividir a avaliação da plataforma em duas visões que serão apresentadas nas próximas seções.

5.1 Avaliação de projeto e interface

Como o Codestand é uma plataforma desenvolvida para a própria comunidade científica, faz sentido que, após entrar em modo de produção, ela pudesse ser submetida à avaliação dos seus membros, apesar de a usabilidade e a interface refletirem algumas características herdadas do Datatracker e dos *mockups* pré-definidos no início do projeto.

Porém, como não está concluído, ainda não é possível fazer uma avaliação muito precisa com relação aos aspectos funcionais da plataforma. Mesmo sendo uma demanda solicitada pela própria comunidade, por enquanto é difícil mensurar se o que está feito era exatamente o esperado e se reflete as necessidades da organização. Mas, no estágio atual, é possível analisar os principais pontos positivos e negativos tanto na metodologia do projeto, como na implementação e na interface. Além da própria análise que a equipe de desenvolvimento pode fazer a respeito da plataforma, alguns desenvolvedores do Datatracker e entusiastas da área tem sido consultados para que estes possam dar suas opiniões de forma mais isenta.

Portanto, nesta seção, será feita uma análise baseada nos erros e acertos do projeto, com relação à visão do autor da presente monografia e de alguns comentários dos membros do próprio IETF.

5.1.1 Pontos positivos

Através do *feedback* recebido pela lista de discussão da ferramenta, e de algumas demonstrações já realizadas, alguns pontos têm sido destacados como positivos. Os principais foram:

- Arquitetura - Alguns meses antes de ter escrita sua primeira linha de código, já se discutiam nas listas de desenvolvimento questões relativas à arquitetura. Por ser uma aplicação incorporada de outra, o Codestand herda diversos paradigmas já bem consolidados da arquitetura do Datatracker, o que certamente facilita bastante novos desenvolvimentos. Além disso, conta muito a ferramenta já ter bem traçada a estrutura que a base de dados deveria possuir, o que facilita ao ponto de o desenvolvedor poder se concentrar em escrever os códigos e não ficar refletindo sobre as consequências, conforme elas aparecem.
- Lista de tarefas - Ao início do projeto foi definida a lista de tarefas em três fases, sendo que, na primeira, esta possuía todos os aspectos funcionais básicos para ser posto em desenvolvimento - este é o estágio em que se encontra hoje. As demais etapas contém funcionalidades mais avançadas, características desejadas que ainda devem ser implementadas no sistema, sem risco de perda de usuários por falta destes recursos. Seguindo a dinâmica dos métodos agéis, ter os objetivos traçados e divididos em pequenas tarefas torna o desenvolvimento dinâmico, e o desenvolvedor não fica preso ao mesmo trabalho muito tempo..
- *Mockups* - A interface dos sistemas, quando começou a ser implementada, já estava toda desenhada em *mockups*, totalmente planejada, e, portanto, bastava apenas ser integrada ao código Django. Assim, como no caso da base de dados, ter em mente o que se espera da interface permite que o desenvolvedor acelere a implementação e implantação, de forma que rapidamente veja o resultado.

Estes pontos, conciliados com o fato de saber da importância da plataforma antes mesmo de ela ser desenvolvida, e de poder demonstrá-la aos clientes finais, também certamente possibilitou que o projeto se mantivesse sempre na direção correta. Isto porque, uma vez que algo não tivesse sido bem entendido ou corretamente traçado, certamente haveria maiores retrabalhos ao final do projeto.

Do ponto de vista técnico, os grandes acertos do projeto - na visão do autor - foram certamente a aplicação de um MVC que permitiu que módulos fossem reusados e a sepa-

ração dos ambientes de desenvolvimento. Um dele foi os ambiente de desenvolvimento, onde pode-se inserir novas características, enquanto no ambiente de produção mantém-se uma versão estável, de modo que os membros do IETF podem ir testando e reportando erros.

5.1.2 Pontos negativos

Por mais bem definido que um projeto possa estar e tendo ciência de toda sua importância, ainda assim algumas coisas não dão certo. Seja por inexperiência da equipe de desenvolvimento, por imprevistos, ou até mesmo equívocos na sua implementação. Além disso, quando se desenvolve já tendo prevista uma meta e data, às vezes é preciso aceitar alguns *trade-off*. Ou seja, eventualmente é preciso aceitar que algo pode sair um pouco inferior do que o planejado, por ser menos importante no momento. Nesse caso, isso é menos problemático do que o normal, pois aqui ainda é possível consertar o que for necessário em um ponto futuro do projeto. Em outras circunstâncias, no entanto, pode-se apenas guardar os erros como aprendizado e tentar situações diferentes no futuro.

Dentre os pontos de destaque negativo do Codestand, podem se destacar os seguintes:

- Baixo auxílio gráfico a novos usuários - Mesmo considerando que a interface foi um dos pontos positivos, por contas dos *mockups*, quando membros que não sabiam do funcionamento do Codestand eram questionados, diversas críticas se dirigiram às nomenclaturas utilizadas e na navegação. Isso ocorreu, provavelmente, por haver pouca ajuda em forma de descrições e auxílios gráficos. Foi, basicamente, um problema prático do projeto, que foi deixado um pouco de lado em detrimento de outras funcionalidades que deveriam estar prontas para a data final.
- Apenas um desenvolvedor - Ter somente um desenvolvedor para codificar acarreta em alguns problemas. O fato de haver somente uma pessoa, que não tem para quem passar o conhecimento e a tomada de decisões do nível de implementação, contrasta diretamente com os paradigmas do IETF, onde a pluralidade das decisões é sempre fundamental. Apesar de estar no Github, o código em que a maioria da equipe de gerência possui conhecimento técnico de implementação para discutir, houve uma dependência da disponibilidade de tempo do codificador e uma restrição técnica, à medida que os conhecimentos se restringem aos do mesmo.

Por se tratar de uma plataforma feita em parceria com uma instituição acadêmica, no caso a Universidade Federal do Rio Grande do Sul, é difícil cobrar foco total no projeto, como se obteria em uma empresa privada, por exemplo, onde a qualidade do projeto está diretamente relacionada ao lucro da mesma.

Na universidade, porém, algo que deve sempre ser levado em conta, mais até mesmo que os acertos são justamente a aprendizagem que o projeto proporcionou.

5.2 Avaliação do Desempenho

Outro viés que, certamente, pode ser abordado com relação à avaliação do Codestand são as análises de desempenho da plataforma, tanto em respostas às consultas do banco de dados como da própria implementação. Isso porque as análises, feitas durante o desenvolvimento, foram pontos cruciais para a implantação da plataforma, e para que esta pudesse ser, de fato, usada pelo mundo todo.

Para entender melhor a avaliação, nessa seção serão retomados alguns pontos da trajetória para que se possa compreender de que forma o desempenho influenciou no andamento do projeto e também quais foram os métodos de mensuração utilizados.

As tarefas de implementação da plataforma ficaram a cargo da equipe de desenvolvimento do lado brasileiro, ou seja, do Instituto de Informática da UFRGS. No princípio, o ambiente de desenvolvimento também foi todo configurado dentro da Universidade. Uma máquina virtual foi criada no Instituto de Informática sob o domínio de *codematch.inf.ufrgs.br*.

Logo na fase de desenvolvimento inicial, toda a base de dados era local, ou seja, era um *dump* (uma cópia dos dados originais) da base de dados real. Porém, conforme explicitado diversas vezes anteriormente, uma das principais características do Codestand é a integração com o Datatracker e principalmente com sua base de dados. Isto porque é através do Datatracker que são feitos os cadastros dos usuários e, deste modo, é extremamente necessário que exista comunicação entre as duas, para que se possa criar uma nova conta no Datatracker, e, logo em seguida, efetuar o credenciamento no Codestand.

Além disso, qualquer documento criado no Datatracker necessita estar automaticamente disponível no Codestand, para que já se dê o passo de captar codificadores - e para isso é de suma importância a conexão entre as aplicações. Portanto, alguns dados importantes do Codestand precisam ser acessados através do Datatracker, e de forma rápida, para não comprometer funcionalidades básicas, como buscar por documentos ou efetuar

o credenciamento no sistema. No entanto, com o Codestand rodando no Instituto de Informática localizado no Rio Grande do Sul, Brasil e o Datatracker, localizado nos Estados Unidos, as consultas trocadas entre as aplicações certamente precisavam atravessar enlaces oceânico. Assim, a integração, que era para ser uma das principais vantagens ao se utilizar a plataforma, ocasionou em recorrentes atrasos durante o uso.

Tão logo mais requisições de código e projetos eram cadastrados na plataforma, o atraso constatado entre as requisições aumentava, de modo a ficar tão grande que a utilização do sistema se tornou impraticável. A solução encontrada, portanto, foi migrar a plataforma do Instituto de Informática para algum serviço de hospedagem localizado Estados Unidos. Portanto, o código foi migrado para um servidor norte-americano, sob o domínio *codestand.ietf.org*. A plataforma em questão escolhida foi a Amazon, e então passou-se a fase de elaboração de análises. De posse das amostragens, seria fundamental atestar então que, rodando nos Estados Unidos, a aplicação escalaria e seria possível a utilização em larga escala da aplicação.

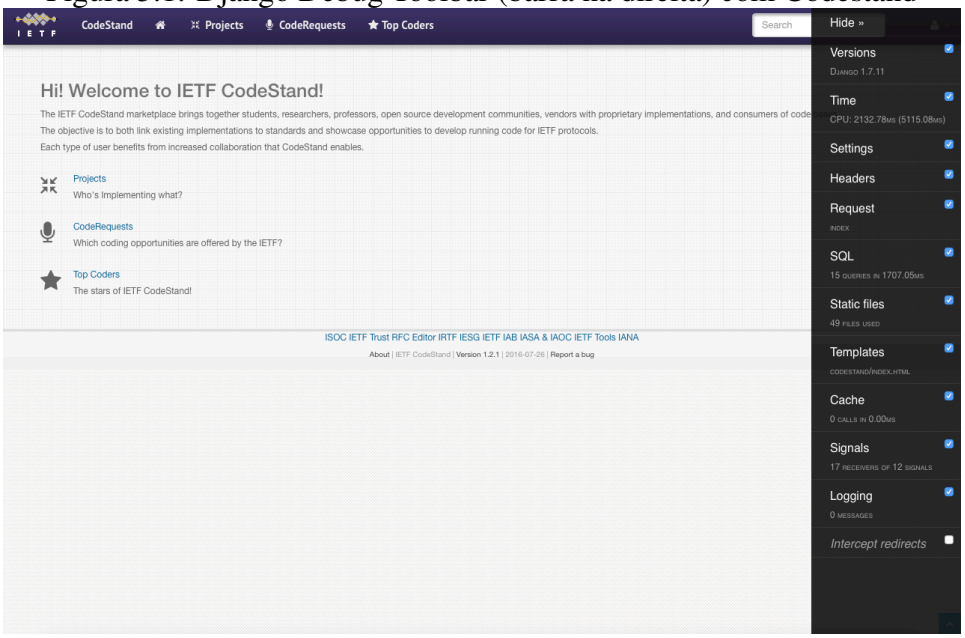
5.2.1 Ferramenta de análise

A ferramenta escolhida para coletar os dados do Codestand foi a Django Debug Toolbar (DJANGO-DEBUG, 2016), uma aplicação específica para desenvolvimento em Django. Com a referida ferramenta, é possível saber exatamente quais consultas foram feitas à base de dados, quantas vezes cada uma foi executada e qual fora o tempo de resposta das consultas. Além disso, foram medidas características como tempo de resposta do código em si e obtidas informações relevantes para a depuração de uma aplicação Django.

5.2.2 Experimentos

Para esta avaliação, foram confrontados dois ambientes, os quais utilizam exatamente o mesmo cenário. Ou seja, os dados cadastrados e o modo de uso do Codestand são exatamente os mesmo. O primeiro ambiente é a máquina virtual localizada no Instituto de Informática da Universidade Federal do Rio Grande do Sul. E o segundo cenário é também uma máquina virtual da Amazon, localizada em São Francisco, na Califórnia, Estados Unidos. A ideia principal é investigar o impacto ocasionado pela eliminação das

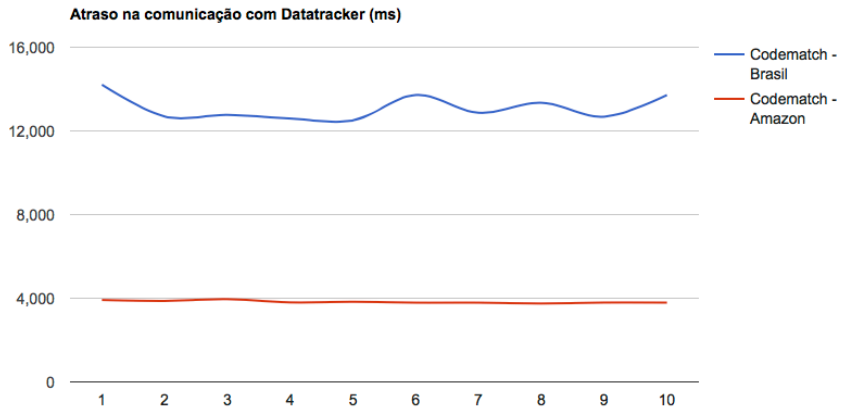
Figura 5.1: Django Debug Toolbar (barra na direita) com Codestand



Fonte: O Autor

consultas transoceânicas. O experimento para amostragem do desempenho consistiu em cadastrar um número grande de projetos e ficar trocando de uma página á outra. Isto porque, quando a lista de projetos é carregada, ela acessa as diferentes tabelas compartilhadas com a base de dados do Datatracker, as tabela de usuários para buscar o codificador de cada projeto e a tabela de documentos para recuperar a área e o grupo de trabalho de cada um dos documentos ligados aos projetos. É importante destacar que o experimento foi rodado dez vezes sobre cada ambiente, de forma intercalada, para minimizar variações em decorrência de instabilidades nos servidores e/ou outros motivos quaisquer.

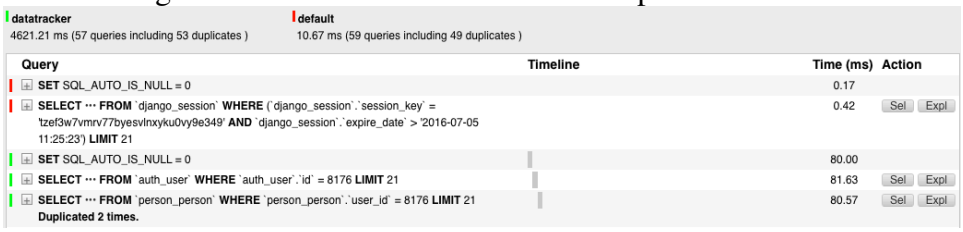
Figura 5.2: Comparação entre Codestand no Brasil e na Amazon



Fonte: O Autor

Analisando a Figura 5.2 é possível notar que existe uma diferença média de quase

Figura 5.3: Atrasos das consultas feitas pelo Codestand



Fonte: O Autor

10 segundos entre o ambiente no Brasil e o ambiente no Estados Unidos, o que, para um usuário de Internet é um tempo considerável de atraso. Na prática, segundo os usuários experimentais, o ganho de tempo devido a migração para a Amazon mudou a experiência de impraticável do primeiro ambiente para natural neste segundo ambiente.

Portanto, é possível verificar que houve uma melhora significativa do domínio *co-dematch.inf.ufrgs.br* para o *codestand.ietf.org*. É por isso, então, que atualmente a plataforma se mantém hospedada no servidor da Amazon, e até o presente momento não houve reclamações com relação a utilização da plataforma, pelo menos no quesito tempo de resposta.

Mesmo com a grande melhora no atraso constatado entre os ambientes, a partir da Figura 5.3 é possível perceber que ainda existem diversas consultas duplicadas. Portanto, um dos próximos passo de otimização do projeto será reduzir essas duplicatas, quando espera-se obter um tempo médio de resposta ainda melhor e, por conseguinte, ter um atraso que nunca implique em problemas de navegação.

6 CONCLUSÃO E TRABALHOS FUTUROS

A presente monografia, realizada como trabalho de conclusão do curso de Ciência da Computação, abordou de forma intercalada aspectos técnicos referentes à implementação do Codestand, aspectos teóricos da Internet e ainda alternativas para permitir que a comunidade evolua. O foco do trabalho é destacar a plataforma Codestand, enfatizando seu modo de funcionamento, sua implementação e os problemas que se propõe a resolver. Porém, para entender corretamente o contexto, o IETF foi apresentado como plano de fundo e, com ele, diversas nuances da principal organização científica da área de redes de computadores.

Alguns resultados pessoais foram atingidos em decorrência do projeto, dentre eles, a capacidade de evoluir e maturar uma plataforma complexa do início ao fim. Ou seja, mais que qualquer projeto desenvolvido em trabalhos de disciplinas ou atividades extracurriculares, a capacidade de planejar, implementar, documentar e expor é o saldo positivo da experiência até aqui. Visto que o Codestand se iniciou como uma bolsa de iniciação científica, o processo de planejamento e implementação ocorreu com naturalidade, devido a disponibilidade e interesse de todos os envolvidos.

O desenvolvimento do projeto proporcionou duas oportunidades de expor o Codestand. O primeiro foi a participação no IETF-95, no qual houve a oportunidade de participar de uma das reuniões do *Internet Engineering Task Force*, o que proporcionou a chance de conhecer alguns dos gerentes envolvidos no Codestand, além de desenvolvedores do Datatracker e do IETF em geral. Durante a reunião, mesmo não tendo havido uma apresentação formal, foi possível trocar experiências e ouvir sugestões que foram fundamentais para que o Codestand esteja atualmente em funcionamento.

A segunda, ocorreu em decorrência de ser uma bolsa de iniciação tecnológica, na qual, existe o requisito de expor o trabalho no FINOVA (FINOVA, 2016), ao final de um ano de bolsa. Como o FINOVA é uma feira tecnológica permite equiparar o trabalho com outros nas mais diversas áreas. Visto que não houve grandes problemas na exposição, tomou-se como indicativo de que o projeto estava no caminho certo e, portanto, maduro o suficiente para ser dissertado como trabalho de conclusão.

Certamente, durante o quase um ano e meio que o Codestand vem sendo desenvolvido, ele trouxe inúmeros aprendizados. Como explicitado ao longo da monografia, mas, provavelmente os erros sejam mais significantes, uma vez que os percalços implicam em tentar não repetir o ponto de falha, e mais, é preciso encontrar o caminho correto para

o futuro. Mas, claro, tanto os pontos positivos e negativos colhidos do projeto são um grande legado, ao passo que são ensinamentos que poderão ser passados aos próximos desenvolvedores, como destacados no capítulo de Avaliação.

Na Avaliação, também foi feito um paralelo entre os positivos e negativos destacados pelos usuários experimentais e um balanço do autor com relação ao estado atual da plataforma. Porém, é importante reafirmar alguns valores realmente positivos que ficam como legado. Dentre eles, destaca-se a capacidade da equipe de desenvolvimento de traçar metas e esboçar a arquitetura e a interface antes de começar a codificar qualquer coisa. Com relação aos pontos que devem ser melhorados para a sequência do Codestand, ou para qualquer outro, evidencia-se a necessidade de uma equipe de desenvolvedores, para que haja uma maior troca de experiências.

Visto que o planejamento inicial do Codestand conta com três fases de desenvolvimento e este trabalho de conclusão foi realizado considerando apenas a primeira etapa concluída, naturalmente espera-se que, no futuro, outras etapas sejam, aos poucos, incorporadas. Para isso, é necessário que a fase experimental em que a plataforma se encontra hoje seja realmente um sucesso e o projeto possa se expandir, aumentando o número de mentores, codificadores e interessados em ajudar.

No nível do desenvolvimento, as próximas etapas são desenvolvimento da barra de pesquisa no topo da plataforma, que atualmente não funciona, a criação de um *dashboard* (painel), que é espécie de minha página existente em diversas plataformas, além de prover mecanismos para aumentar a comunicação entre mentores, codificadores e administradores da plataforma. Outras pendências, obviamente, são corrigir os pontos fracos enumerados no capítulo 5. Alguns passos já vem sendo feitos em paralelo ao desenvolvimento da presente monografia, mas como não é possível atualizar este trabalho diariamente, considerou-se um momento específico para ser descrito.

Portanto, em termos da implementação e implantação da plataforma Codestand, o que foi considerado é o equivalente ao apresentado no FINOVA, em setembro de 2016, sendo este dado momento como o estágio que seria aqui dissertado. Espera-se que, de agora em diante, o Codestand não só evolua em termos práticos, atingindo seus objetivos traçados e aproximando mais a comunidade de Redes ao redor do mundo, como possa gerar novas apresentações, trabalhos científicos e principalmente que sirva ao IETF na sua missão de tornar a Internet cada vez melhor e mais forte.

REFERÊNCIAS

ALVESTRAND, H. T. **A Mission Statement for the IETF**. [S.l.], 2004. 1-7 p. Available from Internet: ⟨http://www.rfc-editor.org/rfc/rfc3935.txt⟩.

APACHE. 2016. Available from Internet: ⟨https://www.apache.org/⟩.

BOOTSTRAP. 2016. Available from Internet: ⟨http://getbootstrap.com/⟩.

BRADNER, S. **Key words for use in RFCs to Indicate Requirement Levels**. [S.l.], 1997. 1-3 p. Available from Internet: ⟨http://www.rfc-editor.org/rfc/rfc2119.txt⟩.

BRADNER, S. The internet engineering task force. In: **Open Sources: Voices from the Open Source Revolution**. [S.l.: s.n.], 1999. p. 48–52.

CODEMATCH-GITHUB. 2016. Available from Internet: ⟨https://github.com/wpjesus/codematch⟩.

CODESTAND. 2016. Available from Internet: ⟨https://codestand.ietf.org⟩.

DARPA; USC, U. of S. C. **Transmission Control Protocol**. [S.l.], 1981. 1-85 p. Available from Internet: ⟨http://www.rfc-editor.org/rfc/rfc793.txt⟩.

DATATRACKER. 2016. Available from Internet: ⟨https://datatracker.ietf.org/⟩.

DJANGO. 2016. Available from Internet: ⟨https://www.djangoproject.com/⟩.

DJANGO-DEBUG. 2016. Available from Internet: ⟨https://django-debug-toolbar.readthedocs.io⟩.

FINOVA. 2016. Available from Internet: ⟨https://www.ufrgs.br/sedetec/?page_id=2113⟩.

FROEHLICH, F. E.; KENT, A. Arpanet, the defense data network, and internet. In: 9TH INTERNATIONAL CONFERENCE ON SPATIAL INFORMATION THEORY. **Encyclopedia of Telecommunications: Volume 1**. [S.l.]: CRC Press, 1991. p. 348–369.

GITHUB. 2016. Available from Internet: ⟨https://github.com/⟩.

HOFFMAN, P.; BRADNER, S. **Defining the IETF**. [S.l.], 2002. 1-4 p. Available from Internet: ⟨http://www.rfc-editor.org/rfc/rfc3223.txt⟩.

HOFFMAN, P.; HARRIS, S. In: **The Tao of IETF - A Novice's Guide to the Internet Engineering Task Force**. [S.l.: s.n.], 2006. p. 1–50.

IETF-1. 1986. Available from Internet: ⟨https://www.ietf.org/proceedings/01.pdf⟩.

IETF-94. 2015. Available from Internet: ⟨https://www.ietf.org/meeting/94/⟩.

IETF-95. 2016. Available from Internet: ⟨https://www.ietf.org/meeting/95/⟩.

IETF-96. 2016. Available from Internet: ⟨https://www.ietf.org/meeting/96/⟩.

MYSQL. 2016. Available from Internet: ⟨http://www.mysql.com/⟩.

PIKE, J. **Defense Data Network**. 2011. Available from Internet: ⟨http://www.fas.org/irp/program/disseminate/ddn.htm⟩.

POSTEL, J. **User Datagram Protocol**. [S.l.], 1980. 1-3 p. Available from Internet: ⟨http://www.rfc-editor.org/rfc/rfc768.txt⟩.

PYTHON27. 2016. Available from Internet: ⟨https://www.python.org/download/releases/2.7/⟩.

RAND. 2013. Available from Internet: ⟨http://www.ipglossary.com/glossary/reasonable-and-non-discriminatory-rand-terms/⟩.

RFC-INDEX. 2016. Available from Internet: ⟨https://www.ietf.org/download/rfc-index.txt⟩.

SCRUM. 2016. Available from Internet: ⟨https://www.scrumalliance.org/why-scrum⟩.

SQL-INJECTION. 2016. Available from Internet: ⟨http://www.w3schools.com/sql/sql_injection.asp⟩.