# Visão

Trata-se de um sistema de biblioteca online dedicado aos pesquisadores do Laboratório de Psicologia Anomalística e Processos Psicossociais da USP[[1]](#footnote-1). A biblioteca deverá conter artigos e livros completos bem como resumos de artigos (abstract) e material produzido pelo próprio grupo de pesquisa.

A ideia é que o sistema proposto armazene grande quantidade de resumos, artigos e livros e esteja disponível sem custos para qualquer pessoa interessada.

## Objetivos

* O sistema tornará mais fácil que um pesquisador encontre material relevante ao seu trabalho.
* Maior divulgação pública das pesquisas realizadas nesta área de estudos.
* Espaço para divulgação de material didático produzido pelos pesquisadores.

# Considerações Técnicas

A USP atualmente não disponibiliza ao laboratório os recursos computacionais necessários para hospedagem deste projeto. A estratégia adotada é criar e hospedar a aplicação em outro lugar e entrar com um pedido para homologação da aplicação e provimento dos recursos necessários para sua hospedagem. Caso o aplicativo não seja homologado, seguirá hospedado em outro lugar.

As universidades públicas não permitem que sejam criados subdomínios que apontem para máquinas que não sejam mantidas dentro da estrutura da universidade. Para driblar esta restrição sem infringir as regras e manter o acesso da aplicação dentro de um domínio da USP, optou-se por criar uma aplicação cliente-servidor[[2]](#footnote-2) onde o cliente da aplicação é codificado dentro de um HTML estático hospedado dentro da USP e o servidor fica hospedado em um servidor externo.

Para possibilitar que o cliente seja codificado em HTML estático o aplicativo será criado como SPA (*single page application*)[[3]](#footnote-3) e será servido como um artigo do Wordpress[[4]](#footnote-4) que é fornecido pela universidade para o laboratório.

O projeto não dispõe de verba para sua execução ou manutenção de forma que a hospedagem externa deve se manter com baixo custo para poder ser provisoriamente bancado pelos pesquisadores membros do laboratório. Requisitos modestos de hospedagem também aumentam as chances de que a aplicação seja eventualmente hospedada pela USP.

Visando manter maior flexibilidade para hospedagem do sistema, serão usadas apenas tecnologias abertas que possam ser executadas tanto em ambiente Linux quanto Windows.

# Tecnologias selecionadas para o projeto

## Cliente

Dado o requisito de ser uma aplicação SPA, foi escolhido o *framework[[5]](#footnote-5)* AngularJs[[6]](#footnote-6) para o desenvolvimento do cliente web. Foram consideradas as versões 1 e 2 do framework e, no momento da análise, o angularjs 2 (chamado simplesmente de Angular) ainda não estava maduro suficiente para ser utilizado em projetos.

The Angular 2.0 API is currently in active development and not production ready. This page showcases a preview of proposed methods to help further the discussion in the development community. If you're building a production app today, please use Angular 1.X.[[7]](#footnote-7)

Para facilitar a compatibilidade com diversos navegadores e estar melhor preparado para o future, optou-se por codificar a aplicação angularJs em Typescript[[8]](#footnote-8) pois a linguagem é compilada para javascript e permite utilizar recursos propostos no ECMAScript 6[[9]](#footnote-9), codificar aplicações fortemente tipadas que por sua vez permitem o uso de algumas ferramentas sintéticas que ajudam na validação e codificação do projeto:

Types enable TypeScript developers to use highly-productive development tools and practices: static checking, symbol-based navigation, statement completion, and code refactoring. [[10]](#footnote-10)

Para definição de estilos será utilizado o bootstrap[[11]](#footnote-11). Para garantir melhor compatibilidade entre diversos navegadores além de facilidade de uso, o CSS será codificado em LESS.[[12]](#footnote-12)

Na versão do HTML que será enviada para o wordpress, os arquivos javascript e css estarão minificados, ficarão condidos dentro de um bundle[[13]](#footnote-13) e estarão embutidos no código html da SPA. O mesmo acontece com as views do angular que serão embutidas no html principal utilizando o gulp-html2js[[14]](#footnote-14). Estas rotinas de otimização serão executadas pelo gulp[[15]](#footnote-15) utilizando as bibliotecas gulp-asset-manifest[[16]](#footnote-16), gulp-minify-css[[17]](#footnote-17) e gulp-uglify[[18]](#footnote-18). As

## Armazenamento de dados

Por se tratar de uma biblioteca virtual, é muito imperativo que a aplicação disponibilize opções avançadas de busca e para tal fim, é recomendado que os dados da biblioteca sejam armazenados em um sistema especializado em buscas

A maioria dos sistemas de busca disponíveis gratuitamente é baseada na engine lucene[[19]](#footnote-19) e, por isso, oferecem recursos bastante semelhantes diferindo apenas na facilidade e conveniência de uso. Para este projeto foram considerados o Apache Solr[[20]](#footnote-20), Elastic Search[[21]](#footnote-21) e o Stario Cassandra[[22]](#footnote-22).

O Stratio Cassandra foi descartado por não possuir recurso de *facet search*[[23]](#footnote-23)*[[24]](#footnote-24)* e algumas comparações entre o Solr e Elastic Seatch foram analizadas[[25]](#footnote-25)[[26]](#footnote-26) para decidir qual sistema de busca seria mais conveniente para o projeto. O elastic search foi escolhido por sua maior facilidade de instalação e uso.

Após definir o Elastic Search como sistema de busca a ser utilizado neste projeto, foi realizado um levantamento para avaliar se haveria necessidade de armazenagem dos dados em uma outra base de dados ou se o elastic search atenderia todos os requisitos para armazenamento permanente de dados.

Diversas referências apontaram que o sistema não deve ser utilizado como base permanente de dados com destaque para o artigo “Call me maybe: Elasticsearch 1.5.0”[[27]](#footnote-27) que realiza uma avaliação extensiva do aplicativo e conclui:

My recommendations for Elasticsearch users are unchanged: store your data in a database with better safety guarantees, and continuously upsert every document from that database into Elasticsearch. If your search engine is missing a few documents for a day, it’s not a big deal; they’ll be reinserted on the next run and appear in subsequent searches. Not using Elasticsearch as a system of record also insulates you from having to worry about ES downtime during elections.

Finally, Elastic has gone from having essentially no failure-mode documentation to a delightfully detailed picture of the database’s current and past misbehaviors. This is wonderful information for users and I’d love to see this from other vendors.

Para o armazenamento permanente de dados foi avaliado se a melhor estratégia seria utilizar uma base SQL ou NO-SQL , tendo em vista que o maior acesso a dados será realizado pelo Elastic Search, optou-se pela conveniência de um SQL tradicional , maior facilidade de hospedagem e melhor desempenho[[28]](#footnote-28).

Foram avaliados os 2 sistemas de banco de dados open-source mais populares: MySQL e Postgres e após leitura de algumas análises realizadas por terceiros, optou-se pelo uso do MySQLpor se tratar do sistema mais popular para hospedagem

MySQL is integrated into several hosting packages. The “LAMP” (Linux, Apache, MySQL, and PHP) standard is a part of most hosting packages. This means that when you order hosting on a Linux server, the host probably uses MySQL as its customers’ database solution. This is another reason why it’s probably best to learn MySQL, because you will run into its use more often when ordering hosting solutions.[[29]](#footnote-29)

MySQL não é sempre ACID (necessita algumas configurações específicas)[[30]](#footnote-30)[[31]](#footnote-31) e comum relatarem problemas estranhos com o MySQL[[32]](#footnote-32)[[33]](#footnote-33) mas, por usarmos um sistema especializado para busca, devemos nos limitar a consultas simples no MySQL e, nestas circunstâncias, nenhum problema deve ser esperado.[[34]](#footnote-34)

## Servidor

A aplicação servidor será desenvolvida seguindo a arquitetura REST (Representational State Transfer)[[35]](#footnote-35) visando aproveitar as vantagens de escalabilidade e performance que este modelo proporciona além de se tratar de um padrão fácil de manter e com bom suporte nas mais diversas tecnologias existentes.

Para a troca de dados entre cliente e servidor, será utilizado o formato JSON (JavaScript Object Notation) que é um formato enxuto, fácil de ler e suportado de forma universal.[[36]](#footnote-36)

Para a codificação do servidor REST foram consideradas as linguagens C#[[37]](#footnote-37) (.NET), Java[[38]](#footnote-38), Python[[39]](#footnote-39), Javascript (Node.Js[[40]](#footnote-40)) e Elixir[[41]](#footnote-41).

### Linguagem

O C# é a linguagem que a equipe possui maior experiência em desenvolvimento e provavelmente seria a escolha que proporcionaria desenvolvimento mais rápido da aplicação. Apesar desta importante vantagem, descartou-se o uso desta linguagem pelo fato do framework que a acompanha ter sido transformado recentemente em código aberto e ainda estar em fase de suporte ao Linux[[42]](#footnote-42). Apesar de existir o framework mono como implementação alternativa do .NET na plataforma[[43]](#footnote-43), esta possui alguns detalhes de implementação distintos do framework oficial sobretudo na implementação do módulo system.web[[44]](#footnote-44)

O Java foi descartado por não ter ciclo de desenvolvimento lento e burocrático e aparentemente não ser mais utilizado para desenvolver aplicações web modernas.[[45]](#footnote-45)[[46]](#footnote-46)

Elixir apareceu como uma boa alternativa para desenvolver aplicações web leves e escaláveis. Testes realizados indicaram que ele possui um dos melhores desempenhos disponíveis para aplicações distribuidas[[47]](#footnote-47) além de possuir um modelo de gestão de processos bastante interessante[[48]](#footnote-48). A linguagem foi descartada por possuir sintaxe e paradigma distintos do que a equipe já está acostumada e a curva de aprendizado desta linguagem traria incertezas e riscos ao projeto.

Para decidir entre o uso de Python e do Node.js foi realizada uma extensiva pesquisa na internet. Foram encontrados relatos de vários times que migraram de python para node.js[[49]](#footnote-49)[[50]](#footnote-50), quadros comparativos entre as duas linguagens[[51]](#footnote-51) e algumas críticas ao Node.js[[52]](#footnote-52)[[53]](#footnote-53), comparações entre Node.js e o .NET[[54]](#footnote-54)[[55]](#footnote-55). Apesar das comparações serem geralmente favoráveis ao node.js, muitos desenvolvedores Node.js estão migrando para o Go[[56]](#footnote-56). Incluindo o criador de um dos principais frameworks para desenvolvimento de aplicações web[[57]](#footnote-57). Ao procurar pelos possíveis motivos desta migração, foi encontrado um artigo que explica em maiores detalhes alguns dos problemas do node.js[[58]](#footnote-58).

Tendo em vista o encontro não planejado com a linguagem Go, foi feita uma nova pesquisa para validar se esta é uma linguagem adequada às necessidades do projeto e alguns testes de desempenho mostraram que ela possui bom desempenho para interfaces REST[[59]](#footnote-59)[[60]](#footnote-60) e possui várias bibliotecas que facilitam a criação deste tipo de serviço.[[61]](#footnote-61)

Adotaremos o Go como linguagem para o projeto mesmo levando em conta que para uma eventual instalação nos servidores da USP será necessário um esforço provavelmente maior do que o necessário para instalar uma aplicação equivalente desenvolvida em python.

### Web Framework

Para facilitar e acelerar o desenvolvimento da aplicação REST, será utilizado um framework para desenvolvimento web. Dada a não-familiaridade com a linguagem e o curto prazo do projeto foi realizada uma pesquisa procurando pelas bibliotecas mais populares que se adequam as necessidades de um projeto.[[62]](#footnote-62)[[63]](#footnote-63) [[64]](#footnote-64) O uso de uma biblioteca mais popular faz com que a maioria dos problemas eventualmente encontrados possam ser resolvidos apenas buscando por problemas semelhantes na internet além de garantir um mínimo de qualidade da biblioteca e de sua documentação

Dentre os frameworks mais populares pesquisados, optamos por utilizar o Gorilla[[65]](#footnote-65).

### Injeção de Dependência

Visando maior facilidade na execução de testes, foi feito um levantamento sobre qual ferramenta para injeção de dependência[[66]](#footnote-66) seria utilizada no projeto e verificou-se que dadas as características da linguagem, não é necessário realizar injeção de dependência para execução de testes unitários bastando utilizar o conceito de interfaces já presente na linguagem.[[67]](#footnote-67)

### Teste unitários

Testes unitários[[68]](#footnote-68) são peças fundamentais para produção de um software de qualidade e apesar da biblioteca padrão da linguagem oferecer suporte a testes unitários[[69]](#footnote-69), o uso de outras bibliotecas é recomendado para facilitar a elaboração de testes em cenários mais complexos. Para estes casos complementares, será utilizada a biblioteca testify.[[70]](#footnote-70)

### Logging

Logs são muito úteis para identificar e resolver problemas de um software, após breve pesquisa[[71]](#footnote-71) foi decidido que o logrus[[72]](#footnote-72) será utilizado como ferramenta de log do projeto.

### Caching

Embora espera-se um número modesto de acessos na aplicação a ser desenvolvida, a utilização de um sistema de cache[[73]](#footnote-73) pode ajudar a aliviar os requisitos computacionais necessários para implantação da aplicação. Será utilizado o golang-lru[[74]](#footnote-74)

### Documentação

Outra parte integrante de um software de qualidade é a documentação de código, a ferramenta padrão é o GoDoc[[75]](#footnote-75) e será utilizada no projeto.

## Estilo e Nomenclatura

Para os arquivos javascript, utilizaremos o padrão de estilo recomendado pela W3C[[76]](#footnote-76). Para os arquivos go, utilizaremos os padrões sugeridos pelos criadores da linguagem.[[77]](#footnote-77) Para validação automática destes padrões utilizaremos o SublimeLister-jslint[[78]](#footnote-78) e o golint[[79]](#footnote-79) integrado ao GoSublime[[80]](#footnote-80)

## Ambiente de Desenvolvimento

Como editor de textos será utilizado o Sublime Text 3[[81]](#footnote-81), para versionamento de arquivos será utilizado o TortoiseGit[[82]](#footnote-82) e para comparação de arquivos e gerenciamento de merges será utilizado o KDiff3[[83]](#footnote-83). Todas estas ferramentas são gratuitas para uso não-comercial.

Todo desenvolvimento será realizado em uma máquina Windows 10.

# Versionamento do Sistema

Git

Sistema de branches

# Gerenciamento de tarefas

TFS / visual studio online

Scrum

Sprints

# Infraestrutura

Cliente

Server rest

Mysql

Elastic search

1. <http://www.ip.usp.br/portal/index.php?option=com_content&view=article&id=2544%3Ainter-psi-laboratorio-de-psicologia-anomalistica-e-processos-psicossociais&catid=384%3Ainter-psi&Itemid=211&lang=pt> [↑](#footnote-ref-1)
2. <https://pt.wikipedia.org/wiki/Cliente-servidor> [↑](#footnote-ref-2)
3. <https://en.wikipedia.org/wiki/Single-page_application> [↑](#footnote-ref-3)
4. <https://wordpress.com> [↑](#footnote-ref-4)
5. <https://pt.wikipedia.org/wiki/Framework> [↑](#footnote-ref-5)
6. <https://docs.angularjs.org/misc/faq> [↑](#footnote-ref-6)
7. <https://angular.io/docs/js/latest/api/> acessado em 29/07/2015 [↑](#footnote-ref-7)
8. <https://en.wikipedia.org/wiki/TypeScript> [↑](#footnote-ref-8)
9. <https://en.wikipedia.org/wiki/ECMAScript#Harmony.2C_6th_Edition> [↑](#footnote-ref-9)
10. <http://www.typescriptlang.org/> acessado em 29/07/2015 [↑](#footnote-ref-10)
11. <http://getbootstrap.com/> [↑](#footnote-ref-11)
12. <http://lesscss.org/> [↑](#footnote-ref-12)
13. <https://msdn.microsoft.com/en-us/magazine/dn451436.aspx> [↑](#footnote-ref-13)
14. <https://github.com/fraserxu/gulp-html2js> [↑](#footnote-ref-14)
15. <http://gulpjs.com/> [↑](#footnote-ref-15)
16. <https://www.npmjs.com/package/gulp-asset-manifest> [↑](#footnote-ref-16)
17. <https://www.npmjs.com/package/gulp-minify-css> [↑](#footnote-ref-17)
18. <https://www.npmjs.com/package/gulp-uglify> [↑](#footnote-ref-18)
19. <https://lucene.apache.org/> [↑](#footnote-ref-19)
20. <http://lucene.apache.org/solr/> [↑](#footnote-ref-20)
21. <https://www.elastic.co/products/elasticsearch> [↑](#footnote-ref-21)
22. <https://github.com/Stratio/stratio-cassandra> [↑](#footnote-ref-22)
23. <https://en.wikipedia.org/wiki/Faceted_search> [↑](#footnote-ref-23)
24. <https://github.com/Stratio/stratio-cassandra/blob/master/doc/stratio/extended-search-in-cassandra.md> [↑](#footnote-ref-24)
25. <https://www.youtube.com/watch?t=42&v=S1Md3LDJPLs> [↑](#footnote-ref-25)
26. <http://solr-vs-elasticsearch.com/> [↑](#footnote-ref-26)
27. <https://aphyr.com/posts/323-call-me-maybe-elasticsearch-1-5-0> [↑](#footnote-ref-27)
28. <http://developer.olery.com/blog/goodbye-mongodb-hello-postgresql/> [↑](#footnote-ref-28)
29. <https://blog.udemy.com/mysql-vs-postgresql/> [↑](#footnote-ref-29)
30. <https://en.wikipedia.org/wiki/ACID> [↑](#footnote-ref-30)
31. <http://www.rdx.com/Blog/dba_tips/2010/11/mysql-is-not-acid-compliant/> [↑](#footnote-ref-31)
32. <http://developers.slashdot.org/story/15/03/19/1930226/why-i-choose-postgresql-over-mysqlmariadb> [↑](#footnote-ref-32)
33. <https://dev.mysql.com/doc/refman/5.0/en/bugs.html> [↑](#footnote-ref-33)
34. <http://www.quora.com/What-are-pros-and-cons-of-PostgreSQL-and-MySQL> [↑](#footnote-ref-34)
35. <http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm> [↑](#footnote-ref-35)
36. <http://json.org/json-pt.html> [↑](#footnote-ref-36)
37. <https://en.wikipedia.org/wiki/C_Sharp_(programming_language)> [↑](#footnote-ref-37)
38. <https://en.wikipedia.org/wiki/Java_(programming_language)> [↑](#footnote-ref-38)
39. <https://www.python.org/> [↑](#footnote-ref-39)
40. <https://nodejs.org/> [↑](#footnote-ref-40)
41. <http://elixir-lang.org/> [↑](#footnote-ref-41)
42. <http://news.microsoft.com/2014/11/12/microsoft-takes-net-open-source-and-cross-platform-adds-new-development-capabilities-with-visual-studio-2015-net-2015-and-visual-studio-online/> [↑](#footnote-ref-42)
43. <http://www.mono-project.com/> [↑](#footnote-ref-43)
44. <http://go-mono.com/status/status.aspx?reference=4.5&profile=4.5&assembly=System.Web> [↑](#footnote-ref-44)
45. <http://programmers.stackexchange.com/questions/102090/why-isnt-java-used-for-modern-web-application-development> [↑](#footnote-ref-45)
46. <http://programmers.stackexchange.com/a/152248> [↑](#footnote-ref-46)
47. <https://gist.github.com/omnibs/e5e72b31e6bd25caf39a> [↑](#footnote-ref-47)
48. <http://elixir-lang.org/getting-started/processes.html> [↑](#footnote-ref-48)
49. <https://journal.paul.querna.org/articles/2011/12/18/the-switch-python-to-node-js/> [↑](#footnote-ref-49)
50. <http://geekforbrains.com/post/why-im-switching-from-python-to-node-js> [↑](#footnote-ref-50)
51. <https://docs.google.com/spreadsheets/d/1SZCThPcHv9SaPfDXL-aQxbCvuNS4n0UMSsClQvtdvn8/edit#gid=0> [↑](#footnote-ref-51)
52. <http://notes.ericjiang.com/posts/751> [↑](#footnote-ref-52)
53. <https://medium.com/@ivanguardado/node-js-the-bad-parts-b4b9d23ef7e2> [↑](#footnote-ref-53)
54. <http://www.haneycodes.net/to-node-js-or-not-to-node-js/> [↑](#footnote-ref-54)
55. <https://thomasbandt.com/is-nodejs-better-than-aspnet> [↑](#footnote-ref-55)
56. <http://thenewstack.io/from-node-js-to-go-why-one-startup-made-the-switch/> [↑](#footnote-ref-56)
57. <https://medium.com/@tjholowaychuk/farewell-node-js-4ba9e7f3e52b> [↑](#footnote-ref-57)
58. <https://medium.com/@shijuvar/web-development-trends-for-2015-and-beyond-c2d3c1ef5718> [↑](#footnote-ref-58)
59. <https://speakerdeck.com/stanaka/lightweight-rest-api-server-by-go-and-performance-compalison-with-implementations-of-other-languages> [↑](#footnote-ref-59)
60. <https://www.linkedin.com/grp/post/3712244-5874534047023132673> [↑](#footnote-ref-60)
61. <http://thenewstack.io/a-survey-of-5-go-web-frameworks/> [↑](#footnote-ref-61)
62. <http://thenewstack.io/a-survey-of-5-go-web-frameworks/> [↑](#footnote-ref-62)
63. <https://changelog.com/on-go-web-application-ecosystem/> [↑](#footnote-ref-63)
64. <http://www.alexedwards.net/blog/a-mux-showdown> [↑](#footnote-ref-64)
65. <http://www.gorillatoolkit.org/> [↑](#footnote-ref-65)
66. <http://martinfowler.com/articles/injection.html> [↑](#footnote-ref-66)
67. <http://www.captaincodeman.com/2015/03/05/dependency-injection-in-go-golang/> [↑](#footnote-ref-67)
68. <http://artofunittesting.com/definition-of-a-unit-test/> [↑](#footnote-ref-68)
69. <http://golang.org/pkg/testing/> [↑](#footnote-ref-69)
70. <https://github.com/stretchr/testify> [↑](#footnote-ref-70)
71. <https://www.reddit.com/r/golang/comments/28vh81/robust_logger/> [↑](#footnote-ref-71)
72. <https://github.com/Sirupsen/logrus> [↑](#footnote-ref-72)
73. <https://en.wikipedia.org/wiki/Cache_(computing)> [↑](#footnote-ref-73)
74. <https://github.com/hashicorp/golang-lru> [↑](#footnote-ref-74)
75. <http://godoc.org/golang.org/x/tools/cmd/godoc> [↑](#footnote-ref-75)
76. <http://www.w3schools.com/js/js_conventions.asp> [↑](#footnote-ref-76)
77. <https://golang.org/doc/effective_go.html> [↑](#footnote-ref-77)
78. <https://github.com/devdoc/SublimeLinter-jslint> [↑](#footnote-ref-78)
79. <http://github.com/golang/lint/golint> [↑](#footnote-ref-79)
80. <https://github.com/DisposaBoy/GoSublime> [↑](#footnote-ref-80)
81. <http://www.sublimetext.com/3> [↑](#footnote-ref-81)
82. <https://code.google.com/p/tortoisegit/> [↑](#footnote-ref-82)
83. <http://kdiff3.sourceforge.net/> [↑](#footnote-ref-83)