* + 1. **Candidato – Vacancies**

Nesta pagina um candidato será capaz de verificar todas as vagas disponíveis atualmente a que ainda não se aplicou, obtendo todas as instancias de Vacancy que não tem uma instancia de Application associada a vaga e ao candidato atual.

A lista de vagas obtidas poderão ser alteradas em função do desejo do candidato, sendo que o mesmo pode realizar uma pesquisa inteligente sobre as vagas. Esta pesquisa segue um padrão especifico, primeiro é obtido todas as vagas cujo titulo de trabalho seja o escolhido pelo utilizador, criado uma lista de vacancyPresent

Esta lista de vagas são depois filtradas por dois filtros, no primeiro é removido todas as vagas que não incluem pelo menos uma ferramenta selecionada, logo de seguida é realizado um filtro semelhante, mas com linguagens selecionadas. Para ambos os casos é realizado um ciclo que obtém todas as possíveis vagos, para cada vaga é obtidas todas as ferramentas ou linguagens. Estas são também listadas é caso uma for uma ferramenta ou linguagem selecionada, então a vaga é adicionada a lista de possíveis vagas. Devido a ordem como os filtros são aplicados é dado prioridade ao titulo do trabalho, depois as ferramentas e por últimos as linguagens.

Com as vagas disponíveis, um candidato pode aplicar-se a vaga, o que envolve criar uma instancia da entidade Application e outra da ApplicationCurrentStep, com ambas relacionado a vaga escolhida ao candidato que a escolheu.

* + 1. **Colaborador – Application**

Na Figura 32 é demonstrado como cada passo duma candidatura permite um conjunto de ações dependentemente do estado e tipo do passo. Todas estas ações são de responsabilidade de outra pagina, todas exceto três a “Continue”, a “Hire” e a “End”. A primeira permite passar a candidatura para o próximo passo do processo de entrevista, já o segundo contrata o candidato e altera a vaga como necessário e por ultimo terminar a candidatura.

A ultima é a mais simples, a instancia da Aplication que representa a candidatura é terminada e é enviado um email ao candidato a informar que a sua candidatura fui terminada.

A ação “Hire” é um quanto mais complexa, a mesma começa por remover todas as entrevistas marcadas e já realizadas, o que envolve remover qualquer instancia de Event associado as entrevistas ainda a realizar. Com as entrevistas removidas é criado uma nova instancia de Employee, baseada na informação do currículo do candidato a contratar, que é de seguido destruído, o que resulta em todas as instancias que forma o dossier do candidato serem também removidos. Tendo o empregado estabelecido, é revocado o papel de candidato ao candidato e é concedido o papel de empregado.

Anteriormente fui estabelecido que uma vaga inclui o numero de indivíduos necessários para preencher a mesma, sendo que “Hire” envolve o contrato dum candidato para tal vaga o numero é decrementado neste ponto, o que envolve atualizar a instancia de Vacancy que representa a vaga. Depois do decremento, caso o numero de indivíduos chegar a zero, todas as aplicações e entrevistas associadas a vaga são removidas da base de dados, com s candidatos de tais aplicações são informados da terminação da sua candidatura. Por ultimo é de facto removida a vaga da base de dados.

Por ultimo temos a ação “Continue”, começa por atualizar o passo atual, passando o estado do mesmo para “Finished”. De seguida é obtido o passo seguinte da vaga ao passo terminado, que é verificado como geral[[1]](#footnote-1) ou não. Se o passo for geral então é verificado se o candidato já realizou uma entrevista geral, o que se acontecer então o passo é ultrapassado. Se a entrevista geral ainda não ocorreu então o passo é não ultrapassável e já iremos verificar o que ocorre sobre o passo.

Na situação do passo não ser geral, é verificado se o candidato já realizou uma entrevista com o mesmo formulário associado ao novo passo, que se acontecer então tal passo é ultrapassado, obtendo o próximo passo, onde esta verificação também ocorre. Esta verificação também ocorre ao passo seguinte ao passo geral, se o mesmo for ultrapassado. Este ciclo continua obtendo o próximo passo e verificar se o mesmo pode ou não ser ultrapassado.

Quando se chegar ao passo que não pode ser ultrapassado, incluído o passo geral sem entrevista previa, a instancia de CandidateCurrentStep é atualizado com o estado “ToSchedule” e a VacncyStop do passo não ultrapassado. Sendo também enviado um email ao candidato a informar o mesmo do passo, incluído o que ira ocorrer no mesmo.

Algo que deve ser notado é que existe sempre um passo que não pode ser ultrapassado, sendo que o passo de contracto (que existe sempre em qualquer vaga) não tem um formulário associado e por isso o “Continue” não encontra uma entrevista que o permita ultrapassar o passo.

Menus nas paginas

Originalmente os menus eram mais simples, mas também menus eficazes, sendo que nas primeiras versões eram compostos unicamente por botões que serviam de *links*.

Tal estrutura levantou um problema muito cedo e notável, o espaço ocupado por tantos botões. Usando os botões só para incluir os *links* já estabelecidos, cada menu ocuparia três vezes o espaço vertical atualmente ocupado pelo menu.

Uma possibilidade considerada para estes menus foi a utilização do *rich widget* Accordion, que também permite fazer algo semelhante. Mas devido algumas limitações visuais, o Ballon foi escolhido sobre o Accordion.

O menu de colaboradores tem *links* para as seguintes páginas:

* Vacancies, vagas já estabelecidas, permite introduzir uma nova vaga,
* Applications, lista de aplicações de candidatos a vagas já estabelecidas,
* Projects, projetos iniciados, permite iniciar um novo projeto,
  + Nota: é possível criar vagas a partir de um projeto.
* Clients, clientes da PS Tec, permite adicionar um cliente,
  + Nota: é possível criar projetos a partir de um cliente.
* Events, demonstra os eventos em que o utilizador participa, permite introduzir um novo evento,
* Forms, lista de formulários existentes, permite adicionar e alterar formulários,
* Curriculums, demonstra as candidaturas espontâneas que ainda não foram recusadas ou aceitadas,
* Candidates, lista dos candidatos já existentes,
  + Nota: cada candidato demonstrado permite ver as informações gerais, o currículo, dossiê de capacidades, aplicações e até entrevista já realizadas.

Já o menu de candidatos tem links para as páginas:

* General, informação geral do candidato,
* Curriculum, currículo do candidato, permite alterar o mesmo,
* Profile, dossiê de capacidades do candidato, permite alterar o mesmo,
* Tutorial, uma página de tutorial que explica algumas características do currículo e dossiê,
* Applications, aplicações a vagas existentes que o candidato esta aplicado,
* Vacancies, vagas disponíveis que o candidato ainda não se aplicou,
  + Nota: a página permite uma pesquisa inteligente de vagas.
* Availability, com esta página o candidato pode demonstrar a sua disponibilidade para entrevista.

Tabs

Originalmente, antes de se descobrir a existência deste rich widget, foram consideradas duas possibilidades, ambas com limitações notáveis.

A primeira era incluir nas páginas com muita informação um menu extra, no topo da página, com vários botões que demonstram a parte da página desejada. Depois considerou-se passar esse menu para um ponto mais abaixo e usar *ballon* para esconder os botões.

Ambas possibilidades envolviam algum código extra da parte da aplicação, para esconder a parte das páginas que não se quer visualizar era necessário utilizar a característica Visible, associando a mesma a uma variável que era alterada com o pressionar dos botões já referidos.

Esta alteração seria sempre acompanhada por um *ajax refresh* que escondia as partes da página não desejadas e demonstra a parte da página desejada.

Então estas formas de visualização envolviam sempre incluir uma variável extra a página, como também uma Screen Action, elementos desnecessários com a utilização da Tabs.

Create e Cancel

Deve ser notado que estas páginas partilham uma logica, a ideia que até o botão “Create” ser pressionado a instancia não é criada, nem as instancias relacionadas.

Esta lógica tem uma vantagem principal, ao não criar as instancias até ao momento desejado nunca são criadas instancias desnecessárias que são removidas de seguida. Também garante que se um utilizador iniciar o processo de criação, mas não o completar, saindo da página sem a utilização dos botões “Create” e “Cancel”, a instancia incompleta não ocupa espaço na base de dados.

Por outro lado, esta lógica levantará algumas dificuldades no processo de formação das instancias a adicionar.

Candidatura Espontânea

Qualquer candidatura espontânea começa com uma possível candidato a fornecer um nome, email, numero de telemóvel e um currículo. Apesar todos serem obrigatórios só o ultimo envolve uma verificação na Screen Action da candidatura. Para tal é verificado se o currículo é nulo e/ou não pdf. Se um ou dois acontecerem, então é lançado uma exempção que indica o problema.

Caso tal exempção não ocorrer é verificado a possibilidade de duas repetições, primeiro o email e ou telemóvel ser associado a um utilizador já estabelecido, nesse caso é terminado a ação e o erro é demonstrado ao utilizador.

No caso da repetição ocorrer devido a um email e ou telemóvel estar associado a uma outra candidatura espontânea, nesse caso é obtido tal candidatura espontânea e a mesma é alterada. Se nenhuma repetição ocorrer, então é simplesmente gerado uma candidatura espontânea completamente nova.

Independentemente se uma candidatura fui gerada ou alterada, é enviado um email ao candidato que comunica que a candidatura fui realizada.

Salvar Availability

Qualquer candidato consegue fornecer a sua disponibilidade para entrevistas pela IView, produzindo uma instancia de Availability.

Mas a introdução duma nova instancia não é tao simples como simples criar uma nova instancia e adiciona-la a base de dados. Sendo que quando um candidato introduz um novo bloco de disponibilidade, existe três formas de chocar com blocos já existentes.

* O bloco adicionado acorre dentro de um bloco já estabelecido
* O bloco adicionado começa antes de um bloco já existente, e acaba no meio de tal bloco
* O bloco adicionado começa no meio de um bloco já existente, e acaba depois de tal bloco

Por essas raçoes, qualquer adição de bloco temporal, começa por verificar se o bloco choca com qualquer bloco existente. No primeiro caso o novo bloco é simplesmente ignorado, nos outros dois casos o bloco original é alterando, o inicio ou fim (segundo caso e terceiro caso respetivamente) para o mesmo valor mas do novo bloco.

Pesquisa Inteligente de Vagas

A pesquisa inteligente de vagas segue um padrão simples, primeiro é obtido todas as vagas cujo titulo de trabalho seja o escolhido pelo utilizador, criado uma lista de vacancyPresent

(estrutura que inclui a informação mais importante de uma vaga e do projeto para que a vaga fui criada)

Tendo essa lista de vagas de vagas a mesma passa por dois filtros, no primeiro é removido todas as vagas que não incluem pelo menos uma ferramenta selecionada, logo de seguida é realizado um filtro semelhante, mas com linguagens selecionadas.

Para ambos os casos é realizado um ciclo que obtém todas as possíveis vagas, para cada vaga é obtidas todas as ferramentas ou linguagens. Estas são também listadas é caso uma for uma ferramenta ou linguagem selecionada, então a vaga é adicionada a lista de possíveis vagas.

Devido a ordem como os filtros são aplicados é dado prioridade ao titulo do trabalho, depois as ferramentas e por últimos as linguagens.

Profile

Na figura XX foi demonstrado a pagina de dossier de capacidades, que como se pode verificar inclui todas as capacidades do candidato. Estas, em grande parte, são demonstradas por um conjunto de Editable Tabels que permite adicionar elementos a mesma, mas não inclui qualquer mecanismo próprio para escrever ou remover o elemento a base de dados.

Para tal é utilizado um Screen Action para adicionar e outro para remover, cada utiliza um Switch que escolhe as ações a realizar tendo em conta uma input que recebe. No adicionar é sempre verificado a adicionar garantindo que não existe repetições, para tal é utilizado uma Server Action diferente. Caso não for levantado qualquer exceção, o que indica que o elemento a adicionar não é repetido, então é criado ou atualizado o objeto a escrever. Já o remover não inclui qualquer verificação, simplesmente remove o elemento.

O único elemento do dossier de capacidades que não é demonstrado por uma Editable Tabel é as aplicações desenvolvidas, sendo que cada instancia desta entidade é muito extensa para uma só linha de texto. Em vez disso as aplicações são demonstradas com ListRecord, em que cada elemento é composto por um WebBlock. Tal WebBlock demonstra um Edit Record ou um Show Record dependente do bloco estar no modo de edição ou não. Para mudar de estado cada bloco inclui um botão “Edit” que altera o estado e, salva qualquer alteração. Outro botão incluído neste bloco é o botão “Delete” que simplesmente notifica a pagina do dossier. O notificar alerta a pagina do “Delete” e por isso não só remove a aplicação escolhida, como também realiza um refersh na lista de aplicações desenvolvidas para demonstra a remoção da aplicação.

Já para adicionar uma nova aplicação, o botão “Add” invoca uma PopUp onde é possível adicionar as informações da nova aplicação. O PopUp inclui dois botões, o “Cancel” simplesmente fecha o PopUp, já o “Add” adiciona a aplicação a base de dados, notifica a pagina do dossier e fecha a PopUp. Quando a pagina do dossier é notificada da adição duma aplicação realiza um refresh a lista de aplicações para demonstra a nova aplicação.

A estrutura desta pagina passou por algumas alterações, a utilização das tabelas e lista foi sempre comum, mas pequenas alterações a volta destes elementos ocorreram durante o desenvolvimento. Primeiro as tabelas e lista incluíam dois botões, o “Next” e “Previous”, que permitiam alterar a pagina de elementos demonstrados. De facto tais botões eram incluídos em todas as paginas que incluíam uma lista de instancias.

Estes botões envolviam um quanto código extra da parte da aplicação, primeiro era utilizado quatro variáveis, o *current* que representava a página da lista verificada, nRows o numero de elemento por cada página da lista, needsNext e needsPrev que demonstram a necessidade do botão Next e o botão Prev. As duas ultimas variáveis seriam utilizadas para definir as características de visibilidade e habilitar dos botões.

Para verificar a necessidade do Prev era bastante simples, sendo que desde que *current* fosse maior que zero então havia pelo menos uma página anterior a mostrar.

Já a necessidade de Next era mais complexo, primeiro a informação para a listagem teria de ocorrer com um SQLQuery em vez de um Aggregate (como é feito atualmente), e nesta *query* era obtida nRows mais 1 instancia da entidade. Se o resultado dessa *query* for um numero de instancias igual a nRows mais 1 então é necessário o Next, para no mínimo obter a ultima instancia.

No caso desta pagina as variáveis teriam de ser repetidas para cada tabela e lista, exceto nRows, o que envolvia 3 arrays de inteiros o que tornava o processo ainda mais complexo.

Eventualmente este sistema foi substituído por algo mais simples, a utilização do rich widget List\_Navigation. Com a utilização do List\_Navigation estas variáveis são desnecessárias e o *query* pode ser substituído por um Aggregate, que em si é otimizado, dai ter sido escolhida sobre o sistema dos botões Prev e Next. Outro fator desta escolha é o facto que o List\_Navigation permite vários botões numerados, não sendo restinguido aos botões Next e Previous, que também inclui.

Esta substituição infelizmente não fui possível para as Editable Tables, sendo que tal substituição só é possível nas List Record e nas Table Record porque os dois incluem a propriedades “Start Index”, propriedade que o Editable não inclui. Esta propriedade permite escolher o primeiro elemento a demonstra da fonte de informação associada a lista ou tabela.

Por ultimo cada tabela e a lista são incluídos num widget chamado SectionExpandable, este widget permite duas capacidades que facilitam a demonstração das informações. Primeiro esconde a informação até ser selecionado um botão, o mesmo botão esconde a informação quando a mesma é demonstrada. Noutro lado interage com outro richt widget chamado SectionIndex, o que permite saltar para a secção selecionada na mesma.

Application

Na figura XX é demonstrado como cada passo duma aplicação permite um conjunto de ações dependentemente do estado e tipo do passo. Todas estas ações são de responsabilidade de outra pagina, todas exceto três a “Continue”, a “Hire” e a “End”. A primeira permite passar a aplicação para o próximo passo do processo de entrevista, já o segundo contrata o candidato e altera a vaga como necessário e por ultimo terminar a aplicação.

A ultima é a mais simples, a instancia da Aplication que representa a aplicação é terminada e é enviado um email ao candidato a informar que a sua aplicação fui terminada.

A ação “Hire” é um quanto mais complexa, a mesma começa por remover todas as entrevistas marcadas e já realizadas, o que envolve remover qualquer instancia de Event associado as entrevistas ainda a realizar.

Com as entrevistas removidas é criado uma nova instancia de Employee, baseada na informação do currículo do candidato a contratar, que é de seguido destruído, o que resulta em todas as instancias que forma o dossier do candidato serem também removidos. Tendo o empregado estabelecido, é revocado o papel de candidato ao candidato e é concedido o papel de empregado.

Anteriormente fui estabelecido que uma vaga inclui o numero de indivíduos necessários para preencher a mesma, sendo que “Hire” envolve o contrato dum candidato para tal vaga o numero é decrementado neste ponto, o que envolve atualizar a instancia de Vacancy que representa a vaga.

Depois do decremento, caso o numero de indivíduos chegar a zero, todas as aplicações e entrevistas associadas a vaga são removidas da base de dados, com s candidatos de tais aplicações são informados da terminação da sua aplicação. Por ultimo é de facto removida a vaga da base de dados.

Por ultimo temos a ação “Continue”, começa por atualizar o passo atual, passando o estado do mesmo para “Finished”. De seguida é obtido o passo seguinte da vaga ao passo terminado, que é verificado como geral ou não. Para um passo ser geral, o mesmo deve ser uma entrevista geral, entrevista que não ocorrer mais que uma vez para cada candidato. Se o passo for geral então é verificado se o candidato já realizou uma entrevista geral, o que se acontecer então o passo é ultrapassado. Se a entrevista geral ainda não ocorreu então o passo é não ultrapassável e já iremos verificar o que ocorre sobre o passo.

Na situação do passo não ser geral, é verificado se o candidato já realizou uma entrevista com o mesmo formulário associado ao novo passo, que se acontecer então tal passo é ultrapassado, obtendo o próximo passo, onde esta verificação também ocorre. Esta verificação também ocorre ao passo seguinte ao passo geral, se o mesmo for ultrapassado.

Deve ser notado que ultrapassar um salto, envolve adicionar um passo no estado “Finished” a aplicação, assim a entrevista anterior fica acessível pela aplicação atual.

Quando se chegar ao passo que não pode ser ultrapassado, incluído o passo geral sem entrevista previa, é criado uma instancia de ApplicationStep no estado “ToSchedule” e é enviado um email ao candidato a informar o mesmo do passo, incluído o que ira ocorrer no mesmo.

Algo que deve ser notado é que existe sempre um passo que não pode ser ultrapassado, sendo que o passo de contracto (que existe sempre em qualquer vaga) não tem um formulário associado e por isso o “Continue” não encontra uma entrevista que o permita ultrapassar o passo.

CVCheck

Nesta pagina um colaborador pode verificar as candidaturas espontâneas ainda não tratadas e pode executar três ações diferentes. A mais curta das três é “Download”, que utiliza a capacidade Download de Screen Actions, para obter o currículo pdf que o candidato forneceu.

Já a ação “Accept” permite aceitar a candidatura, estabelecendo um candidato com a informação fornecida pela candidatura espontânea. Para tal começa-se por estabelecer uma password, utilizando a função GeneratePassword[[2]](#footnote-2), que é de seguida encriptada utilizando EncryptPassowrd, que encripta o resultado do GeneratePassword com o email do candidato. Note-se que a encriptação é necessária para guarda na base de dados a password.

Tendo a password gerada é criado um utilizador com a informação da candidatura, com email servindo tanto como email do utilizador, como o username do mesmo. Com o utilizador estabelecido é concedido o papel de candidato ao mesmo, como também é estabelecido o CandidateCurriculum com o nome da candidatura e uma instancia de Photo. Para a fotografia do Photo é utilizado o logotipo da PS Tec que é incluído no recursos da aplicação.

Estando estabelecido as informações do utilizador, “Accept” acaba com a destruição da instancia de SpontaneousCurriculum e o enviou de um email, ao candidato a informar o mesmo que a sua candidatura fui aceitada e que mesmo pode aceder o IView com o email fornecido e a password estabelecida ao inicio da ação.

Já “Refuse”, que recusa a candidatura, simplesmente remove a mesma da base de dados, como também envia um email ao candidato, informado o mesmo que a sua candidatura fui ponderada, mas recusada.

Events

Para demonstrar todos os Events em que o colaborador participa, na forma demonstrada na FIGURA XXX, existe uma serie de passos na preparação desta pagina que formam a lista de EventObj, que em si são utilizados para produzir os blocos temporais. Primeiro é obtido os próximos cinco dias uteis, que são obtidos na Server Action FormDates.

De seguida é obtidos todos os eventos que o colaborador participa nos próximos 7 dias, o que envolve realizar uma UNION entre qualquer Event em que o colaborador é o responsável e os Events associados aos EventGroup que incluam o colaborador. Deve ser notado que devido a complexidade deste query é utilizado uma ação SQL em vez de uma Aggregate.

Para demonstrar as varias horas que compõem cada dia, é incluído a Server Action CreateScheduleBlocks, que essencialmente obtém todos os elementos da tabela estática Hours e cria uma lista de ScheduleBlocks. Cada elemento desta lista tem como Start o valor de uma instancia da Hour e como End tal valor mais 30 minutos, utilizando a função AddMinutes.

Por ultimo é necessário produzir a lista de EventObj, o que é realizado pela Server Action EventsToSchedule. Neste é realizado um ciclo duplo, com o primeiro correndo para cada meia hora que forma um dia (instancias de Hour) e o segundo para cada um dos cinco dias que a função recebe (em forma da lista de Date, chamada dates). Este ciclos podem parecer estar fora de ordem, mas devido a como o *widget* List Record funciona, realizar o ciclos ao contrario resultaria em 18 blocos representarem um dia, sendo que 18 é o numero de meias horas que compõem um dia. Na ordem atual um dia é formado por 18 blocos que aparecem numa forma vertical.

Dentro do segundo ciclo é primeiro verificado se existe algum bloco, na lista a devolver, que ocorra na hora e data verificados no ciclo atual. Se tal acontecer é então readicionado tal bloco, assim as informações do bloco são repetidas nos valores de data e hora verificados no ciclo atual.

Na situação de não existir um bloco na lista a ocorrer na data e hora do ciclo atual, então é verificado se existe um evento na lista fornecida a função, cujo inicio e data é igual a hora e data verificadas no ciclo atual. Se tal acontecer um bloco é adicionado a lista a devolver com a informação do evento, caso não acontecer o bloco adicionado é vazio.

EventAddEditGroup

A preparação desta pagina envolve três ações distintas:

* Obter todos os utilizadores participantes do evento, caso a pagina é utilizada para alterar um Event, o que é demonstrado pelo Event Id fornecido ser não null
* Obter todos os utilizadores possíveis, expeto o utilizador atual
* Filtrar os utilizadores obtidos no segundo ponto, removendo os participantes

Para o ultimo ponto é utilizado a Server Action FilterUsersByGroup, que recebe os dois grupos de utilizadores do ponto anterior. Na ação é listado todos os utilizadores participantes e para cada é filtrados a lista de todos os utilizadores, com o resultado da filtração sendo devenido como o produto de retorno da ação.

Deve ser notado que qualquer alteração duma lista afeta a outra, por isso ao remover um utilizador duma lista, o mesmo é adicionado a outra lista, e adicionar um utilizador a uma lista envolve remover o mesmo da outra. Qualquer pesquisa de utilizador a adicionar, envolve o resultado da pesquisa ser filtrado utilizando a FilterUsersByGroup.

Já para escolher o momento em que o evento ira ocorrer envolve uma forma leve da preparação da pagina Events. Para tal todos os eventos que o utilizador atual participa, utilizando uma query semelhante a de Events mas só para um dia.

Mas sendo que o evento só pode ser formado em função da disponibilidade de todos os participantes no grupo, quando os blocos de eventos a considerar têm de considerar todos os participantes.

Para tal é utilizado a Server Action AddEventsByGroup, que verifica todos os utilizadores que lhe é fornecido, para cada é obtido os eventos que o utilizador participa que são adicionados a lista de eventos obtido no ponto anterior. Estes são depois fornecidos a Server Action FillEvents, para formar os blocos de eventos para marcar.

Por ultimo, tendo o grupo e um bloco temporal para o evento, é utilizado a ação CheckSelected para verificar se o bloco fui selecionado pode ser utilizado ou não, que simplesmente envolve verificar se o bloco selecionado já tem um evento associado ou não, se tiver o bloco não pode ser utilizado.

EventAddInterview

Interview

1. Para um passo ser geral, o mesmo deve ser uma entrevista geral, entrevista que não ocorrer mais que uma vez para cada candidato [↑](#footnote-ref-1)
2. Gera um conjunto de caracteres, no caso do “Accept” 15 caracteres, aleatórios. Neste caso utilizando caracteres alfabéticos e não alfabéticos. [↑](#footnote-ref-2)