



Documento de Projeto de Sistema

Social Meet Scheduler

Registro de Alterações:

| Versão | Responsável | Data | Alterações |
|--------|----------------------------|------------|--------------------|
| 1.0 | Ádler Oliveira Silva Neves | 22/09/2019 | Versão Inicial |
| 1.1 | Ádler Oliveira Silva Neves | 22/10/2019 | Correções de erros |

Vitória, ES

2019

1 Introdução

Este documento apresenta o documento de projeto (*design*) arquitetural do sistema Social Meet Scheduler. O sistema busca permitir que usuários registrem encontros geolocalizados, falem com os organizadores do encontro e com seus amigos e recebam um lembrete por email da existência do evento que o usuário marcou presença atribuindo a ele uma estrela 6 horas antes deste. Este documento está organizado da seguinte forma: a Seção 2 apresenta a plataforma de software utilizada na implementação da ferramenta; por fim, a Seção 3 apresenta o projeto da arquitetura de software e suas subseções explicam cada uma de suas camadas.

2 Plataforma de Desenvolvimento

Na Tabela 1 são listadas as tecnologias utilizadas no desenvolvimento da ferramenta, bem como o propósito de sua utilização.

Tabela 1 – Plataforma de Desenvolvimento e Tecnologias Utilizadas

| Tecnologia | Versão | Descrição | Propósito |
|------------------------|--------|---|---|
| Python | 3.7 | Linguagem de programação orientada a objetos e independente de plataforma. | Escrita do código-fonte das classes que compõem o sistema. |
| Django | 2.2.5 | Conjunto de APIs e tecnologias voltadas para a construção rápida de aplicações rápidas, seguras e escaláveis. | Redução da complexidade do desenvolvimento, implantação e gerenciamento de aplicações Web a partir de seus componentes de infra-estrutura prontos para o uso. |
| Django Admin Site | 2.2.5 | Aplicação drop-in que provê um gerenciamento facilitado dos dados do sistema por administradores. | Redução da complexidade de administração do sistema. |
| Django Rosetta | 0.9.3 | Aplicação drop-in que provê um gerenciamento facilitado da tradução de páginas da interface. | Redução do esforço de traduzir o sistema para diferentes idiomas. |
| Django Templates | 2.2.5 | API para a construção de páginas baseadas em templates | Criação das páginas Web do sistema, reutilizando a estrutura visual comum às páginas, facilitando a manutenção do padrão visual do sistema. |
| Django Page Components | 0.1 | API para criação de componentes reutilizáveis | Criação e reutilização de componentes visuais Web de alto nível, componentes customizados de forma a facilitar a manutenção do padrão visual do sistema. |
| Django Forms | 2.2.5 | API para definição e tratamento de formulários | Criação de formulários e seu posterior tratamento, facilitando o desacoplamento das camadas. |
| Django ORM | 2.2.5 | API para persistência de dados por meio de mapeamento objeto-/relacional. | Persistência dos objetos de domínio sem necessidade de escrita dos comandos SQL. |
| PyCDI | 1.1 | API para injecção de dependências. | Integração entre diferentes camadas da arquitetura. |
| PostgreSQL | 11.5 | Sistema Gerenciador de Banco de Dados Relacional gratuito. | Armazenamento dos dados manipulados pela ferramenta. |
| WSGIServer | 0.2 | Servidor de Aplicações e arquivos estáticos para Python WSGI. | Execução das APIs citadas acima em ambiente de desenvolvimento. |
| uWSGI | 2.0.18 | Servidor de Aplicações para Python WSGI. | Execução das APIs citadas acima em ambiente de produção e hospedagem da aplicação Web, dando acesso aos usuários via HTTP. |
| NGINX | 1.17.3 | Servidor de arquivos estáticos e proxy-reverso. | Servir arquivos estáticos e realizar o proxy-reverso no Gunicorn, adicionando a camada TLS em ambiente de produção. |

Na Tabela 2 vemos os softwares que apoiaram o desenvolvimento de documentos e também do código fonte.

Tabela 2 – Softwares de Apoio ao Desenvolvimento do Projeto

| Tecnologia | Versão | Descrição | Propósito |
|--|----------------------------|---|--|
| FrameWeb Editor | 1.0.0. 201908 181134 | Ferramenta CASE do método FrameWeb. | Criação dos modelos de Entidades, Aplicação, Persistência e Navegação. |
| PlantUML | 1.2019.11 | Renderizador de diagramas. | Renderização gráfica de diagramas a partir de texto que o FrameWeb não gera. |
| TeX Live | 2019. 51075-3 | Implementação do L ^A T _E X | Documentação do projeto arquitetural do sistema. |
| GNOME L ^A T _E X | 3.32.0 | Editor de LaTeX. | Escrita da documentação do sistema, sendo usado o <i>template abn-TeX</i> . ¹ |
| Eclipse IDE for Enterprise Java Developers | 4.12.0 | Ambiente de desenvolvimento (IDE) com suporte ao desenvolvimento Java EE. | Ambiente de execução do plug-in do FrameWeb Editor. |
| VSCodium | 1.37.1 | Editor de texto com suporte a extensões com suporte a Python. | Desenvolvimento do código-fonte da aplicação. |
| pip | 19.0.3 | Ferramenta de gerência de dependências de software python. | Obtenção das dependências do projeto. |
| virtualenv | 16.1.0 | Ferramenta para isolar ambientes de desenvolvimento python. | Isolar as dependências do software do restante do sistema. |
| Automake | 1.16.1 | Ferramenta de automação em script | Desenvolver atalhos convenientes para várias tarefas. |

3 Arquitetura de Software

A arquitetura de software do sistema Social Meet Scheduler segue uma arquitetura análoga à padrão sugerida pelo FrameWeb (SOUZA, 2007; SOUZA; FALBO; GUIZZARDI, 2009) baseada no padrão Camada de Serviço (FOWLER, 2002). A Figura 1 ilustra tal arquitetura sugerida e indica onde atuariam os *frameworks* para desenvolvimento Web Java com *JSF*, *JPA*, *CDI*, *PrimeFaces* e *Facelets*.

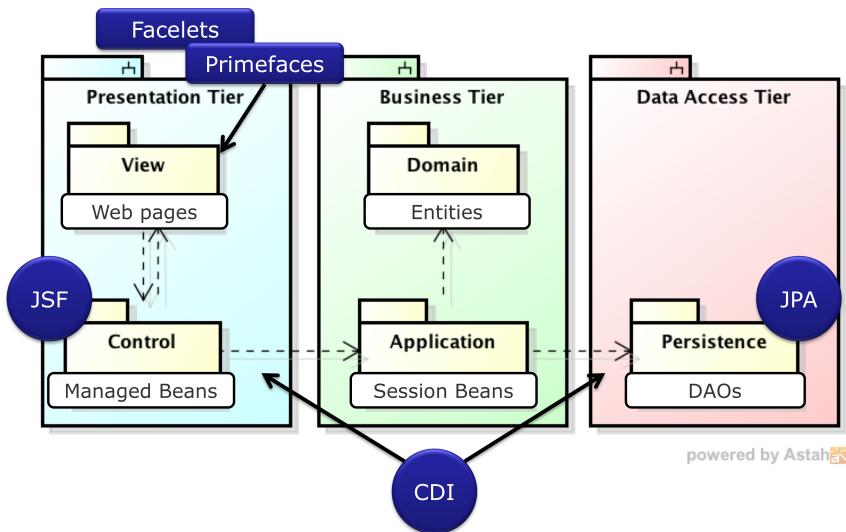


Figura 1 – Arquitetura padrão proposta pelo FrameWeb.

Entretanto, como tal arquitetura não se mantém no padrão Django, que é composta de *models*, *views* e *templates* com responsabilidades diferentes da dos elementos de mesmo nome na arquitetura Java EE. Portanto é necessário fazer aproximações para ambos os modelos dizerem a mesma coisa com palavras diferentes. Se adicionarmos uma camada de serviços entre os *models* e *views*, podemos obter uma arquitetura como a da Figura 2, que é bastante similar a 1. Como Django usa *Active Record* e não *Data Access Objects*, a camada de acesso ao dado foi ignorada.

Nas próximas seções, serão apresentados diagramas FrameWeb relativos a cada uma das camadas da arquitetura do sistema.

3.1 Camada de Apresentação

Como o sistema, ainda no estágio de projeto, ficou grande demais para ser representado em poucos modelos de navegação, cada diagrama desta seção não substitui o outro, mas representa por facetas complementares (não por partes) o sistema.

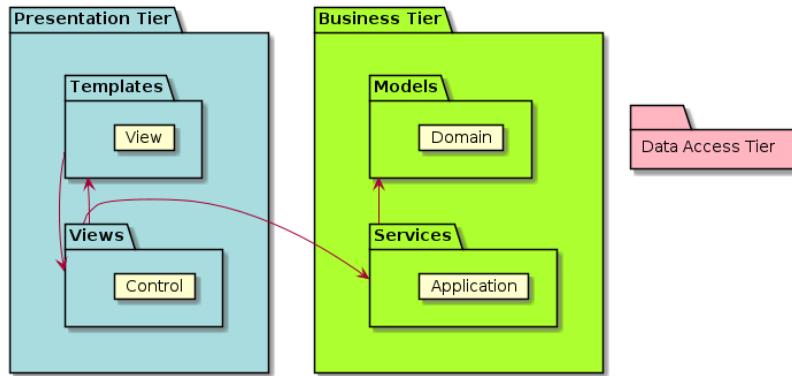


Figura 2 – Arquitetura que busca aproximar o modelo do Django a do FrameWeb.

A Figura 3 traz as classes em seus respectivos pacotes que serão mencionados novamente no decorrer dos próximos parágrafos e figuras, mas tais classes serão mencionadas apenas pelo nome, estarão fora de seus respectivos pacotes e desprovidos de seus atributos e métodos, já que o intuito é melhorar a visualização dos dados através de uma organização mais flexível. Classes adicionais podem aparecer com nomes genéricos, indicando que são parte do *framework* ou de alguma biblioteca; assume-se que sua existência e comportamento são conhecidos e, portanto, irrelevantes modelar em detalhes. Como cada controlador está mapeado a uma rota e o método invocado é o verbo usado no protocolo HTTP, redirecionamentos de URL foram temporariamente representados no modelo como uma página que aciona o método `get` de algum controlador, levando a alguma outra página.

Os controladores do pacote `view` recebem até 3 parâmetros: uma instância de `HttpRequest`, um serviço e uma instância de alguma classe presente no modelo de entidades. A instância de `HttpRequest` contém, dentre várias coisas, o usuário atualmente autenticado no sistema e os parâmetros GET e POST, sendo fornecido pela *framework*. O serviço é injetado por um decorador de uma biblioteca de injeção de dependências. A instância de alguma entidade é injetada por um transformador que, a partir de algum componente de URL nomeado que atua como identificador único, busca o objeto persistido e o insere como parâmetro do método. Todas as requisições são processadas em escopo de requisição, onde apenas alguns poucos dados (identificador de usuário logado, seleção de idioma e seleção de fuso-horário) são armazenados em cookies e variáveis de sessão.

Formulários (pacote `form`) representam apenas os campos do formulário, seu *widget* de apresentação (para uso em templates), seus dados e seus validadores, podendo ser usado em classes que receberiam os estereótipos «page», «controller» e «service», cada um com um diferente propósito: em páginas, exibiria os campos; em serviços, atua como validador e atualizador de entidades; em controladores, facilita o encapsulamento em poucas linhas dos dados de formulário (fornecidos pela instância de `HttpRequest`) e, caso a validação

falhe, a exibição dos erros de validação na mesma página em que o formulário foi editado.

A Figura 4 mostra a navegação de uma página acessível publicamente para uma página que é acessível somente a usuários, mostrando a possibilidade de ser redirecionado para uma tela de login, que, depois de logado, redireciona de volta à página originalmente solicitada. São controladores acessíveis apenas a usuários logados:

- MyAccount
- Friendship (apenas métodos `post` e `delete`)
- Friends
- Conversations
- Conversation
- TalkToFriend
- TalkToMeetOrganizer
- MeetNew
- MeetEdit
- MeetLinksEdit
- MeetLinkEdit
- MeetStars (apenas método `post`)
- MeetStarEdit

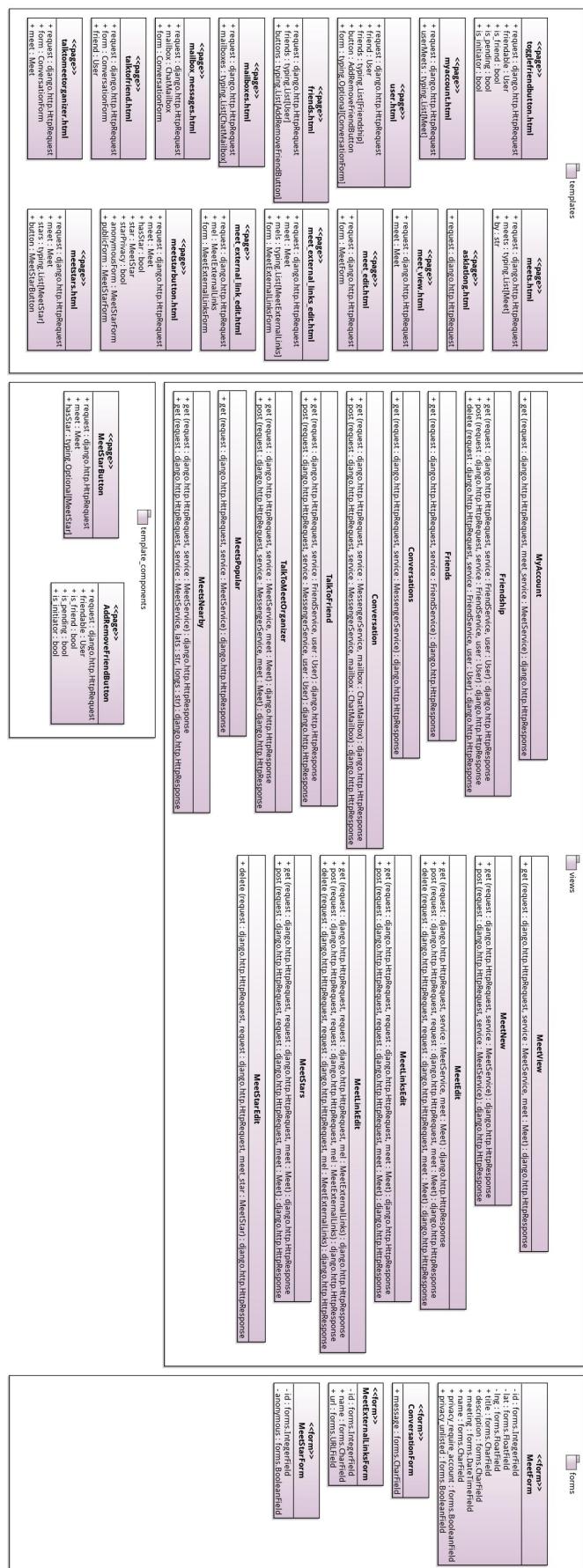


Figura 3 – Modelo de Navegação do Social Meet Scheduler – Apenas classes

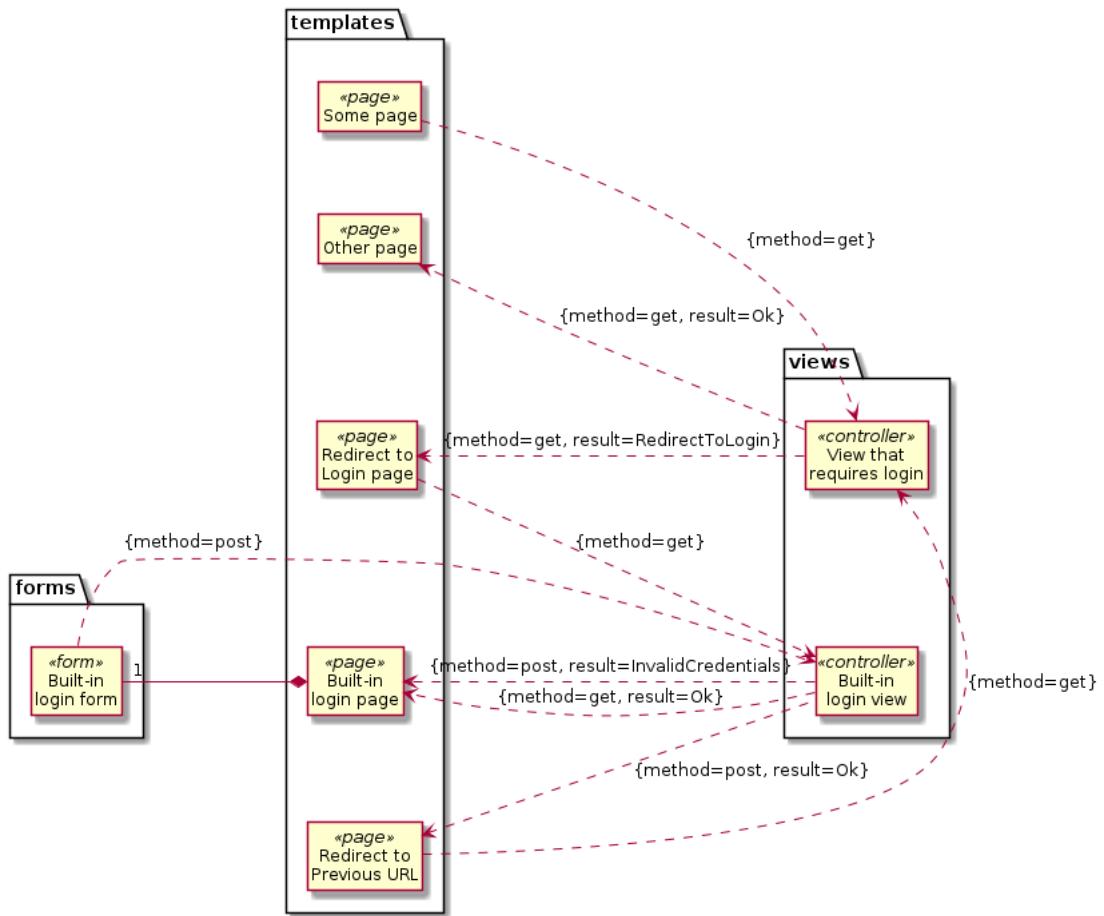


Figura 4 – Modelo de Navegação do Social Meet Scheduler – Login para acessar algum recurso

A Figura 5 mostra a navegação entre duas páginas que possuem autorreferência, e observa-se que o modelo acaba um tanto poluído. Tomando a liberdade para estender o modelo padrão padrão do FrameWeb, adiciona-se o estereótipo «controls» vinculando uma página a um controlador (Figura 6), na qual aquele controlador será controlado apenas por aquele controlador para todos os métodos a menos que explicitamente mencionado o contrário, e setas contíguas no modelo de navegação representam uma requisição ao método `get` do controlador seguido de uma resposta bem-sucedida à página com estereótipo «controlledpage» (Figura 7); considera-se a seta de linha contínua um comportamento explícito. Caso o controlador implemente outros métodos, a Figura 8 ilustra o comportamento padrão esperado. Outro atalho que será introduzido é a seta pontilhada com `result=redirect` de controlador para controlador, que abstrai uma página de redirecionamento, acionando o método `get` do controlador que recebe a requisição (Figura 9). Ainda outra abstração é o método “*”, que significa “qualquer método não explicitamente modelado”.

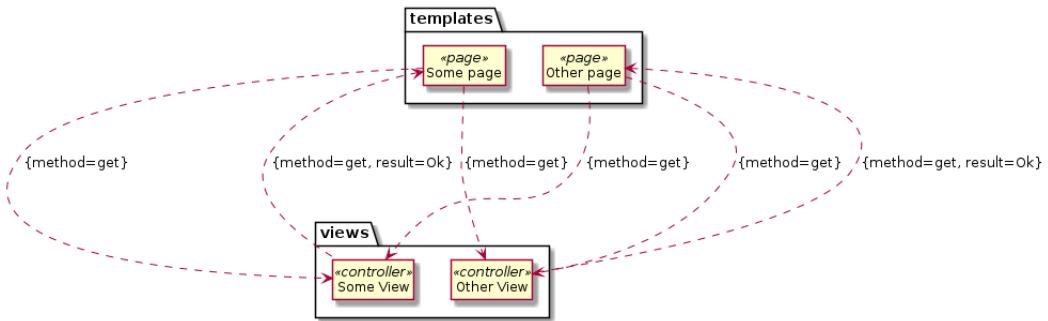


Figura 5 – Modelo que busca justificar a necessidade de estender a linguagem do FrameWeb

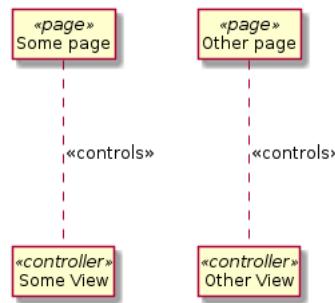


Figura 6 – Atalho proposto para modelo de Navegação do Social Meet Scheduler – Modelo que propõe um atalho, introduzindo o estereótipo «controls»

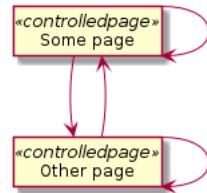


Figura 7 – Atalho proposto para modelo de Navegação do Social Meet Scheduler – Modelo que propõe um atalho, introduzindo a seta cheia e o estereótipo «controlledpage»

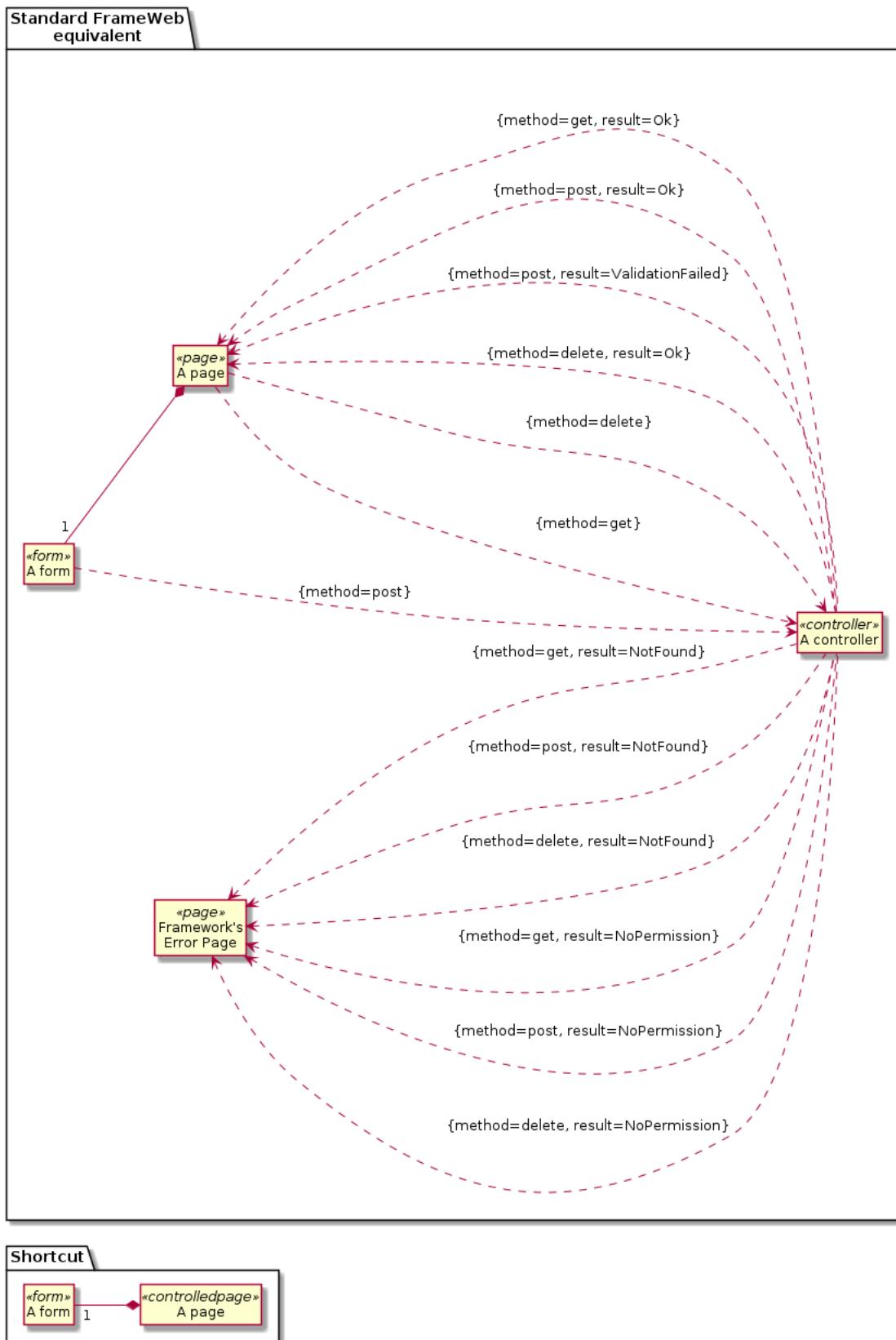


Figura 8 – Atalho proposto para modelo de Navegação do Social Meet Scheduler – Modelo que porpõe um atalho, explicitando todo o ciclo de vida implícito de um «controlledpage» com formulário

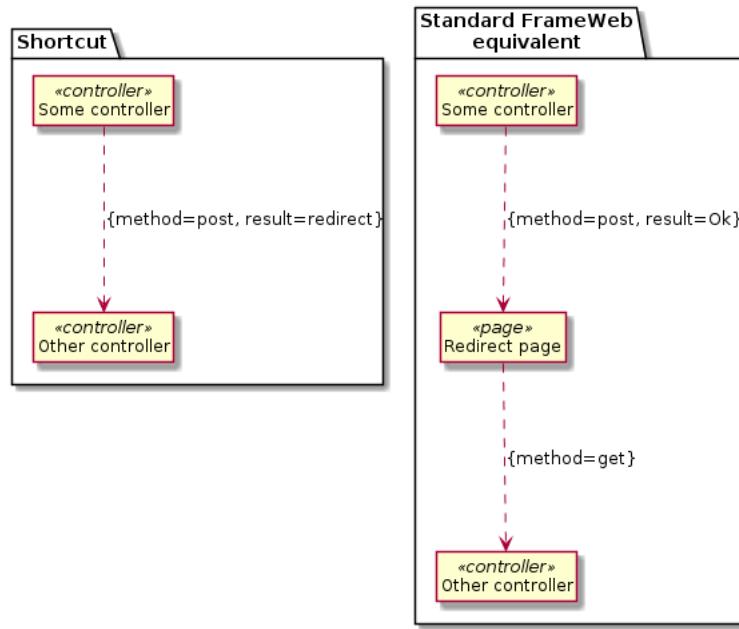


Figura 9 – Atalho proposto para modelo de Navegação do Social Meet Scheduler – Modelo que porpõe um atalho para ligar o redirecionamento de um controlador diretamente no método *get* em outro controlador

Logo, podemos fazer uso de todos os atalhos apresentados acima apresentar as associações na Figura 10 e montar o modelo de navegação na Figura 11.

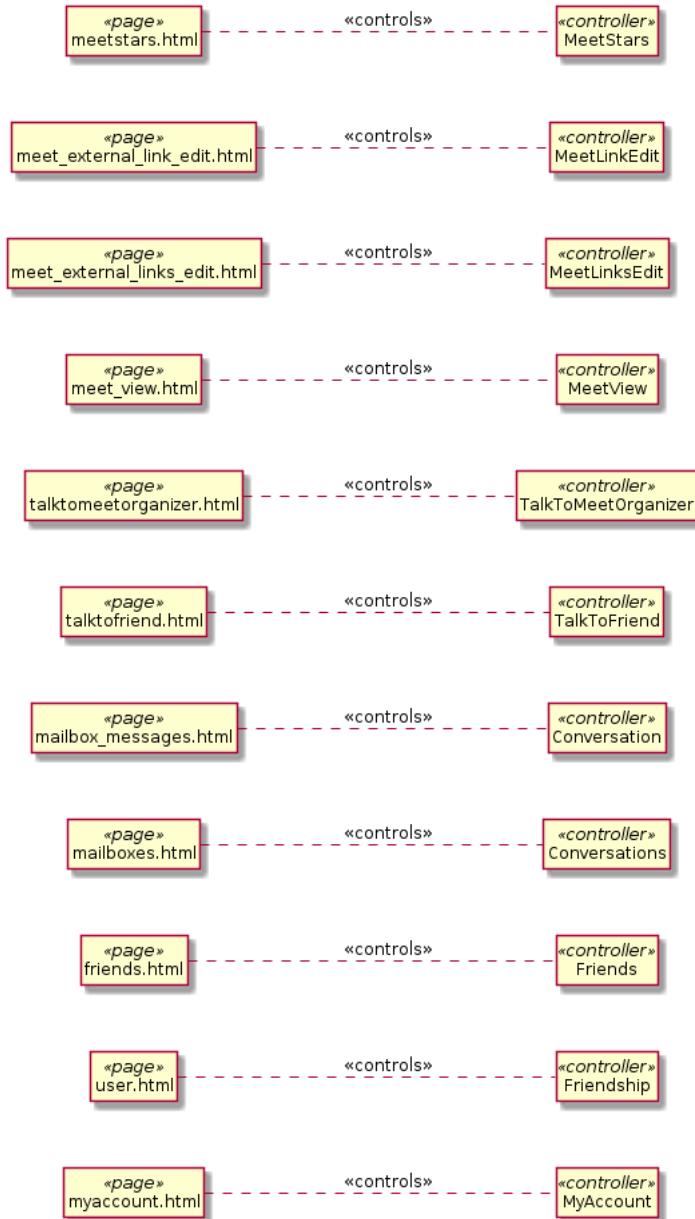


Figura 10 – Modelo de Navegação do Social Meet Scheduler com atalhos – Controladores das páginas

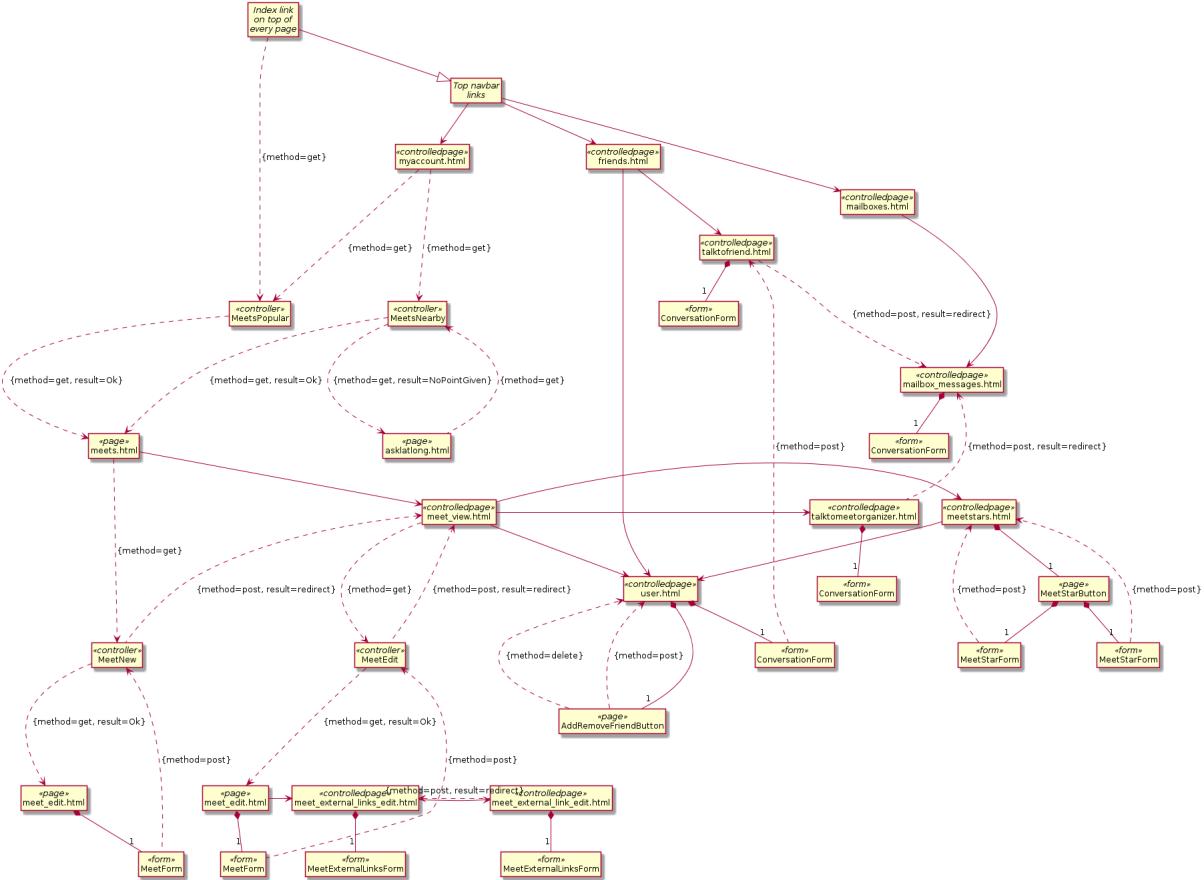


Figura 11 – Modelo de Navegação do Social Meet Scheduler com atalhos – Navegação entre as páginas

Embora tal modelo demonstre o sistema como um todo, alguns detalhes podem se perder no meio de tanta abstração, logo, além da funcionalidade de login descrita na Figura 4, a Figura 12 e a Figura 13 trazem o recorte (sem atalhos) de, respectivamente, enviar uma mensagem para um amigo e de gerenciar os links externos de um encontro.

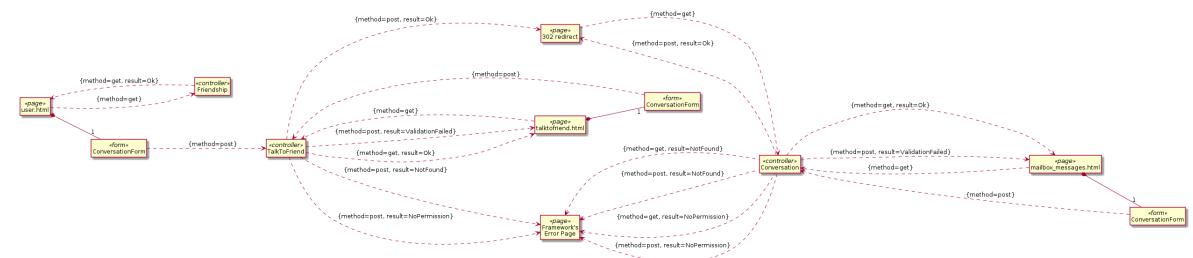


Figura 12 – Modelo de Navegação do Social Meet Scheduler – Enviar uma mensagem para um amigo

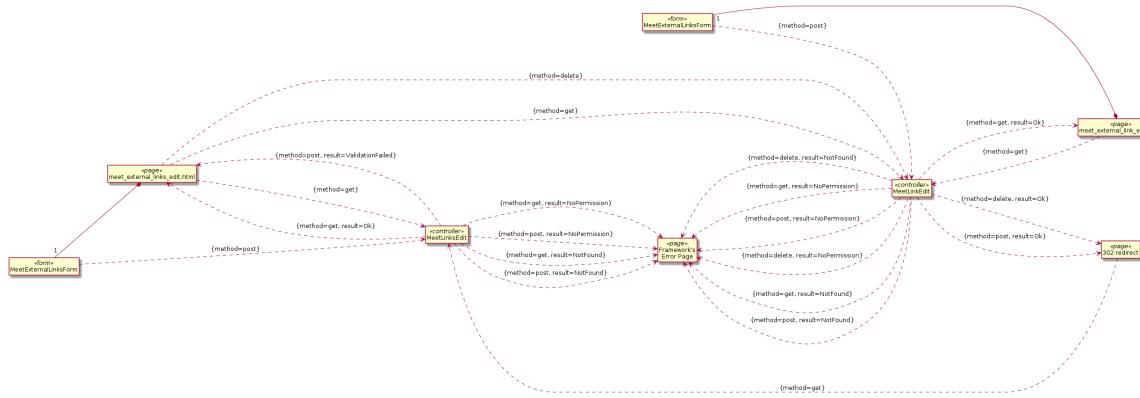


Figura 13 – Modelo de Navegação do Social Meet Scheduler – Gerenciar os links externos de um encontro

3.2 Camada de Negócio

O modelo de entidades do sistema é composto por uma entidade de usuário (`User`) que é implementado pela *framework* e uma coleção de entidades próprias, como expresso no pacote `meet`, que pode ser observado na Figura 14.

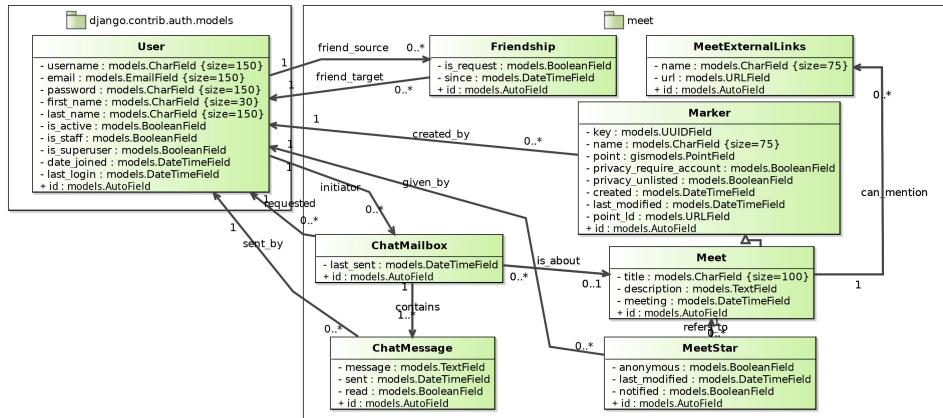


Figura 14 – Modelo de Entidades do Social Meet Scheduler.

A Figura 14 possui algumas particularidades devem ser observadas:

- Todas as classes de modelo herdam de “`django.db.models.Model`”, o qual fornece o atributo “`id`” (colocado explicitamente em todas as classes neste modelo) como chave primária caso nenhuma outra chave primária seja definida, o que é o caso de todas as classes deste modelo; tal superclasse comum também provê mecanismos relativos à persistência, bem como mecanismos de criação, atualização e busca de elementos, abstraindo completamente detalhes relativos à tecnologia de persistência adotada.

- O estereótipo de precisão de datas foi omitido, já que “`models.DateTimeField`” pressupõe implicitamente “`{precision=timestamp}`”, “`models.DateField`” pressupõe implicitamente “`{precision=date}`” e “`models.TimeField`” pressupõe implicitamente “`{precision=time}`” e a inclusão deste apenas dificultaria a visualização do modelo.
- Todas as associações entre entidades possuem navegabilidade bidirecional, na qual a seta apenas indica o sentido de leitura do rótulo da associação.
- Todos os atributos das classes do pacote “`meet`” exceto aqueles decorrentes de associações não podem ser nulos.
- Todas as associações exceto “`is_about`” são representadas por atributos de tipo “`models.ForeignKey`” que não admitem que este armazene nulo.
- A associação “`is_about`” é representada por um atributo de tipo “`models.ForeignKey`” como as outras, mas admitindo valores nulos (devido à cardinalidade `0..1`).
- Uma instância de `User` conhece as instâncias de `Meet` através do atributo de tipo `models.ForeignKey` herdado de `Marker`.
- Uma instância de `ChatMailbox` pode simbolizar tanto uma conversa privada entre amigos (ou ex-amigos) quanto uma conversa entre um usuário e o organizador de um encontro sobre o encontro: se a associação “`is_about`” do lado `ChatMailbox` possuir um `Meet`, é uma conversa entre um usuário e o organizador de um encontro sobre o encontro; caso contrário, é uma conversa privada.
- Uma instância de `ChatMailbox` possui 2 atributos de tipo “`models.ForeignKey`” derivados de associações entre esta entidade e `User` (explicitamente, as associações “`initiator`” e “`requested`”) que armazenam na classe “`ChatMailbox`” quem começou a conversa e quem receberia a primeira mensagem da conversa.
- As classes `Meet` e `Marker` estão separadas em classes diferentes para facilitar estender, futuramente, funcionalidades no sistema.
- A classe `User` é fornecida ou pelo *framework*; está presente no modelo apenas para favorecer a completude .

A camada de aplicação do modelo é expressa pela Figura 15, a qual é redundante com a assinatura dos métodos dos controladores apresentados na Figura 3, mas de uma forma diferente com menos texto.

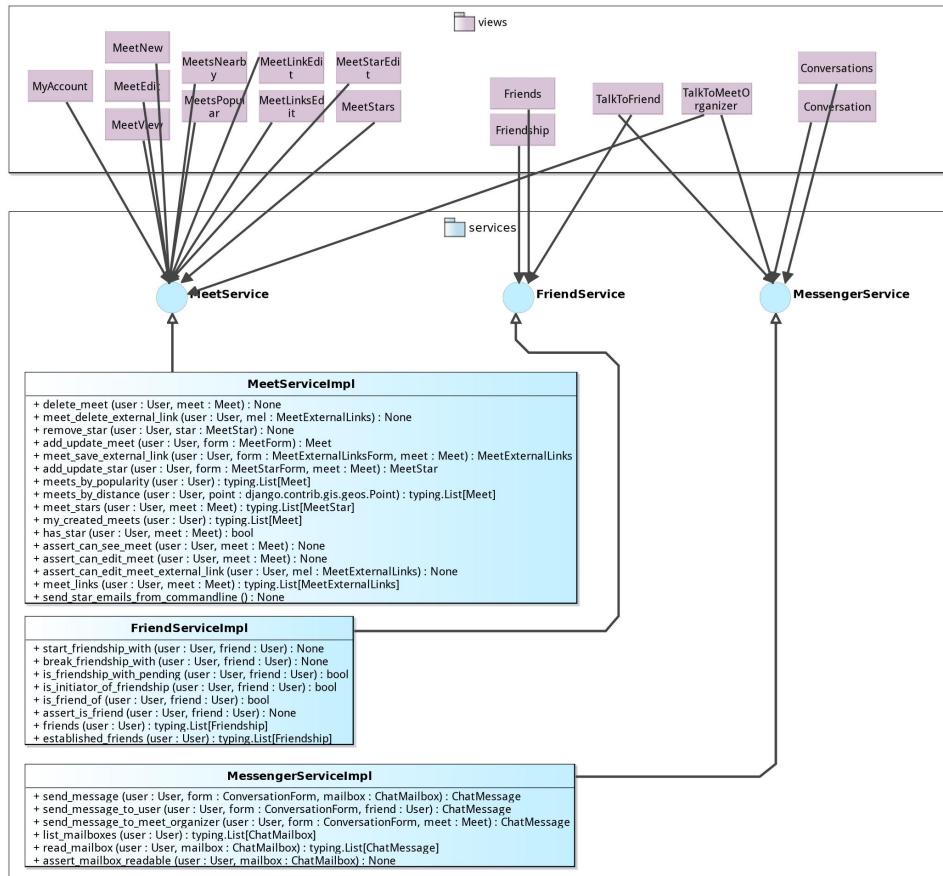


Figura 15 – Modelo de Aplicação do Social Meet Scheduler.

Como Django usa *Active Record* e não *Data Access Objects* e, por conta disso, a camada de acesso ao dado foi ignorada, e, por conseguinte, o modelo de persistência não existir, a Figura 16 explicita quais modelos são gerenciados por quais serviços, suprindo a lacuna causada pela falta deste.

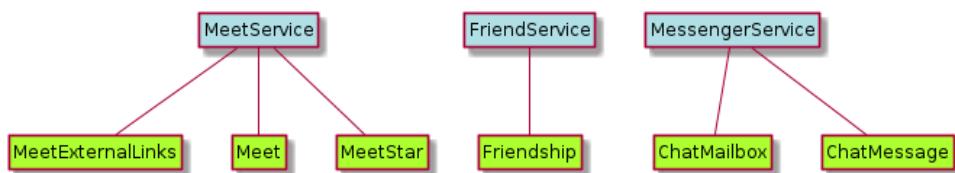


Figura 16 – Modelo que explicita a relação de gerência das entidades pelos serviços do sistema.

3.2.1 Modelo Framework-LD

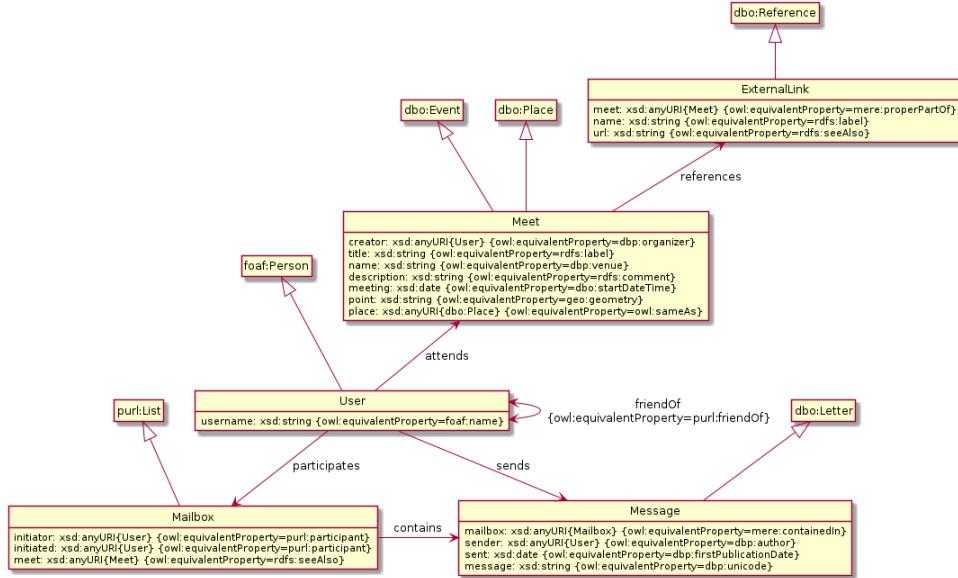


Figura 17 – Modelo de Entidades de *Linked Data* do Social Meet Scheduler.

A Figura 17 mostra a versão *Linked Data* exposta das entidades da Figura 14. A classe ChatMailbox foi exposta como Mailbox. A classe ChatMessage foi exposta como Message. A classe MeetExternalLinks foi exposta como ExternalLink. A classe Friendship foi exposta apenas como uma relação entre duas entidades. A propriedade place da classe Meet da Figura 17 é dada pela propriedade point_id da classe de mesmo nome na Figura 14. A propriedade meet da classe ExternalLink da Figura 17 é dada pela propriedade parent da classe equivalente na Figura 14. De resto, o modelo continua representando os mesmos dados, ainda que usando nomes ligeiramente diferentes e explicitando a presença de algumas propriedades cuja presença era apenas presumida a partir das associações presentes na Figura 14 e mantendo outras ainda presumidas.

Referências

FOWLER, M. *Patterns of Enterprise Application Architecture*. 1. ed. [S.l.]: Addison-Wesley, 2002. ISBN 9780321127426. Citado na página [4](#).

SOUZA, V. E. S. *FrameWeb: um Método baseado em Frameworks para o Projeto de Sistemas de Informação Web*. Dissertação (Mestrado) — Universidade Federal do Espírito Santo, 2007. Citado na página [4](#).

SOUZA, V. E. S.; FALBO, R. A.; GUIZZARDI, G. Designing Web Information Systems for a Framework-based Construction. In: HALPIN, T.; PROPER, E.; KROGSTIE, J. (Ed.). *Innovations in Information Systems Modeling: Methods and Best Practices*. 1. ed. IGI Global, 2009. cap. 11, p. 203–237. ISBN 9781605662787. Disponível em: <<http://www.igi-global.com/reference/details.asp?id=33232>>. Citado na página [4](#).