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

RICARDO CHAGAS RAPACKI

Altea Booking integration into Social Networking

Trabalho de Graduação

Trabalho realizado em Convênio de
Dupla Diplomação com o INP-Grenoble

Porto Alegre, Junho de 2013

Grenoble INP - Ensimag
École Nationale Supérieure d'Informatique et de Mathématiques Appliquées

Projet de fin d'études report

Accomplished in Amadeus London

Altea Booking integration into Social Networking

Ricardo Chagas Rapacki

3^{e} année -— Option ISI

August 2012

Amadeus Services Limited	Internship Manager
World Business Centre Building 3	Amit Gawali
1208 Newall Road	
Middlesex, TW6 WTA	
United Kingdom	Pedagogic Tutor
	Claudia Roncancio

Resumo Extendido

Hoje em dia, com o surgimento das redes sociais, as pessoas estão gradativamente realizando mais tarefas cotidianas em sites como Facebook e Twitter – desde comprar roupas a checar informações sobre o trânsito. Baseado nisto, a Amadeus, empresa líder mundial em soluções de TI para a indústria de turismo e viagem, decidiu desenvolver um aplicativo Facebook para permitir aos usuários buscarem reservas, compartilhá-las e combinar viagens com amigos.

Para isto, as reservas podem ser obtidas através da utilização dos *Web Services* implementados pela Amadeus através de um *record locator* (identificador único de uma reserva) ou criadas sem este identificador ao salvar uma viagem no aplicativo que ainda não foi reservada no sistema.

O núcleo do aplicativo foi implementado na linguagem Python utilizando o *framework* Django, além de diversas outras tecnologias Web como Javascript, jQuery, SOAP, API Facebook, API Google Maps, etc. Suas funcionalidades focam na idéia de administrar viagens já reservadas ou criadas através do aplicativo, compartilhá-las no mural do usuário no Facebook ou convidar diretamente amigos para participar. Quando um usuário é convidado para uma viagem, ele pode utilizar a viagem do amigo como referência e checar, para cada trecho necessário, a disponibilidade de vôos e preços.

A arquitetura da aplicação é dividida nos seguintes elementos: Canvas Facebook, API Facebook, Amadeus We Go App, *Web Services* e banco de dados. O Canvas é um espaço em uma página Facebook onde arquivos HTML, Javascript e CSS do aplicativo são carregados através de um URL e a utilização da API Facebook é essencial para utilizar todas funcionalidades e dados da rede social.

Em relação ao módulo principal da aplicação Amadeus We Go, é utilizada uma arquitetura MVC (Model-View-Controller), onde o arquivo *views.py* controla que dados são apresentados, os *templates* Django nos arquivos HTMLs controlam como estes dados são apresentados e o controlador é o próprio *framework*. Além disso, os dados são salvos em

um banco de dados SQLite, criado pelo Django automaticamente pelo arquivo models.py.

Além deste, há também o banco de dados RFD da Amadeus e o banco de dados acessado pelos *Web Services*. O primeiro é salvo no servidor da companhia e fornece informação atualizada sobre aeroportos, cidades, companhias aéreas, terminais, países e tudo relacionado à indústria de turismo e viagem. Por exemplo, é possível procurar informações de um aerporto através do código IATA de 3 dígitos, como GRU para Garulhos, ou então listar todos aeroportos em determinado país.

A fim de realizar o *deployment* da aplicação, foi necessário configurar um servidor Apache no Linux e não somente utilizar o servidor web disponibilizado pelo Django, pois este último não suporta o protocolo HTTPS nativo, mandatório para aplicações no Canvas do Facebook.

Como já citado anteriormente, três tecnologias são essenciais para o aplicativo Amadeus We Go e serão explicadas mais detalhadamente a seguir. Apesar da API Facebook, por essência, poder ser utilizada normalmente através de requisições HTTP e retornar objetos JSON, o Facebook suporta oficialmente SDKs para Javascript, PHP, iOs e Android para abstrair este processo.

Considerando que Python foi a linguagem de programação escolhida, foi decidido que para abstrair a API seria utilizado o SDK *open-source* chamado Python for Facebook. Este, além de abstrair as requisições HTTP com funções simples de usar, é bastante portável e suportado por uma grande comunidade. Apesar disso, é necessário utilizar o SDK Facebook para Javascript para implementar a autenticação no modo canônico com o Facebook e, em seguida, utilizar o *cookie* obtido no arquivo Python.

Adicionalmente, a API Facebook utiliza o protocolo OAuth 2.0 para autenticação de usuários que fornece um fluxo de autorizações específicas para aplicações web. Uma das maiores vantagens disto é que a aplicação não tem acesso ao usuário e senha porque todo o fluxo é controlado pelo Facebook. O pedido de autorização é feito pela aplicação para o usuário, que é redirecionado a um diálogo Facebook para inserir seus dados e, se correto, o Facebook retorna um *token* de acesso à aplicação.

Este *token* está ligado às permissões definidas pela aplicação no momento da autorização, como acesso a fotos e amigos, e está limitado somente a estas permissões. Deste modo, a aplicação não terá liberdade para fazer nada além do que já pediu e, se definir novas permissões, irá gerar outro diálogo Facebook para criar um novo *token* de acesso.

Para o aplicativo Amadeus We Go, as permissões requisitadas são *user_likes* (para páginas curtidas pelo usuário), *friends_about_me* (para informações pessoais de amigos), *friends_location* (para local de moradia dos amigos) e *publish_stream* (para publicar postagens no mural do Facebook. Esta última permissão é a única que é individualmente revocável pelo usuário, enquanto as outras são obrigatórias para o uso do aplicativo.

Finalmente, o núcleo da API Facebook é a API Graph, cuja função é representar todo grafo da rede social uniformemente com objetos (pessoas, fotos, eventos, páginas, etc) e conexões entre eles (amizades, conteúdo compartilhado, *tags* em fotos, etc). Logo, após adquirir o objeto chamado Graph com a SDK Python for Facebook, qualquer conteúdo pode ser acessado através de seu ID e conexões.

Igualmente importante para o aplicativo, os *Web Services* integram as funcionalidades da Amadeus com a rede social. De modo a obter preços e disponibilidade de vôo atualizados, o aplicativo envia mensagens SOAP para estes *Web Services* da Amadeus e analisa a mensagem que recebe como resposta. Este processo está encapsulado pelo cliente SOAP implementado utilizando a biblioteca nativa urllib2 e é responsável por guardar todas informações necessárias como URL do Web Service, ID da sessão, *token* de segurança e número de sequência. Este último é utilizado para checar se alguma mensagem foi perdida entre cliente e servidor e o incremento do número é realizado por esse cliente.

Além disso, cada serviço utilizado possui um arquivo Python específico que recebe os parâmetros da chamada, cria a mensagem SOAP através de um *template* e analisa a resposta. Entretanto, o primeiro passo para utilizar tais serviços é se autenticar utilizando o LSS (*Logon and Security Server*), um identificador único para cada aplicação ou empregado. Após feito isso, as seguintes operações são utilizadas pelo aplicativo:

- PNR Retrieve by Record Locator: Obtém informações sobre uma reserva ativa, chamada de PNR (*passage name record*) , utilizando um *record locator*
- Flight Availability: Buscar disponibilidade de vôos buscando por países de destino e origem, datas e classe de assento
- Itinerary Fare: Fornece tarifa atualizada para determinado vôo

Finalmente, a última tecnologia a ser detalhada é o banco de dados SQLite, responsável por salvar usuários, passageiros, viagens e ligações entre elas. Adicionalmente, é neste nível que é criada a distinção entre duas reservas diferentes: as já compradas e salvas no sistema da Amadeus e as salvas somente no aplicativo.

Para salvar informações sobre usuários Facebook e passageiros, é utilizada a tabela *Customer.* Uma tupla pode ser criada em dois momentos:

- Quando uma nova viagem é criada, salvando informações do perfil Facebook do usuário para evitar chamar a API e melhorar o desempenho da aplicação
- Quando uma reserva é obtida através de um *record locator,* onde todos passageiros presentes na reserva são salvos sem informações Facebook

No segundo caso, se o usuário em seguida confirme que é um dos passageiros em questão, a tupla é atualizada com suas informações de perfil. Relacionado às viagens, a tabela *Trip* representa as informações sobre uma viagem salva pelo aplicativo e guarda também ligações com viagens de amigos no atributo *linkedTrips*.

Como casos mais simples, estão as tabelas *Booking*, *DummyBooking*, *Trip_Customer*, *Trip_Booking* e *Trip_DummyBooking* para representar, respectivamente, uma reserva presente no sistema da Amadeus, uma reserva feita pelo aplicativo e tabelas para representar as relações entre tabelas. Elas existem, pois, como no caso de Trip_Customer, ela guarda um atributo informando se o usuário é proprietário da viagem ou foi convidado por outro e, se sim, quem é o usuário que convidou.

A última tabela é a *Segment*, representando o trecho de uma reserva, contendo todas informações úteis sobre as viagens. Um segmento está sempre ligado a uma reserva com *record locator* ou uma sem. Logo, se a reserva for destruída, o segmento também é.

De modo a validar cada versão do aplicativo, foi utilizada ferramenta de controle de versionamento distribuído chamado Mercurial. Para isto, é necessário primeiramente que o desenvolver execute um *commit* localmente na máquina e, sempre que a modificação na versão tenha sido significativa, será executado um *nominate* que envia um pedido aos integradores.

Ao receber o pedido, cabe ao integrador a tarefa de baixar o código e executá-lo em sua máquina. Se nenhum problema for encontrado, é executado um *push* e o código atualizado se encontrará no repositório principal, disponível já nas próximas vezes que alguém o recuperar.

Em relação à análise do desenvolvimento do software, a decisão de utilizar Python com Django contribuiu muito para abstrair tarefas como o banco de dados e permitiu criar um código extremamente modular e facilmente extensível.

Por exemplo, caso um novo Web Service da Amadeus venha a ser utilizado, é necessário somente a criação de um novo arquivo com o template da mensagem a ser enviada e a lógica da leitura da resposta, pois a parte da troca de mensagens e exemplos de outros serviços já estão implementados. Além disso, o SDK Python for Facebook foi bastante importante para a integração com a rede social mesmo não sendo oficialmente suportado pelo Facebook.

Por outro lado, houve algumas complicações com a implementação do cliente SOAP. Devido à falta de documentação por parte da própria empresa Amadeus e das bibliotecas testadas, SUDS, SOAPpy e ZSI, o desenvolvimento foi atrasado em um mês e o grande problema era a inconsistência entre a mensagem enviada e a mensagem esperada pelo servidor.

Adicionalmente, o envio de requisições HTTP para os *Web Services* piorou o desempenho do aplicativo e, apesar de uma chamada HTTPS para o API Facebook não ser tão pesada, o conjunto de operações realizadas pode ficar extremamente custoso, pois cresce proporcionalmente em função do número de amigos do usuário, por exemplo.

Considerando possíveis melhorias futuras, a alta latência dos *Web Services* pode ser mitigada se o *deployment* do aplicativo for feito nos servidores do *datacenter* da Amadeus na cidade de Erding, Alemanha, pois o custo de rede será minimizado. Além disto, a utilização de *cache* ou salvar informações no banco de dados pode evitar a chamada de operações pesadas e melhorar o desempenho da aplicação.

Diversas funcionalidades discutidas na fase de especificação do projeto ou outras novas poderiam ser implementadas futuramente para melhorar a experiência do usuário e aproveitar todo o potencial da API Facebook e de todos *Web Services* da Amadeus. Uma das funcionalidades identificada como importante, permitir a compra de passagens aéreas através do aplicativo, não foi implementada devido às regras de negócio da Amadeus. Neste projeto, o aplicativo está limitado a somente salvar as informações sobre as viagens em seu banco de dados.

Concluindo, mesmo que nem todas as funcionalidades planejadas inicialmente tenham sido implementadas, o aplicativo soluciona os problemas que motivaram o desenvolvimento. Adicionalmente, para incluir novas funcionalidades ou atualizar as já existentes, a extensão do aplicativo é simples, seja utilizando novos *Web Services* da Amadeus ou novas funções e dados da API Facebook para maior integração com a rede social. Assim, o projeto pode ser continuado e oferece um futuro promissor como um aplicativo para uso em redes sociais que venha a integrar o cotidiano de viagens e turismo das pessoas.

Abstract

Interested in integrating the services provided by the company with social networks, Amadeus decided to offer an internship opportunity to develop a Facebook application, in which users can retrieve bookings and share them with friends, amongst other functionalities.

The main purpose of the application consists in using Facebook features to enable the users to share trips with their friends and the data stored in the social network to provide useful information in the application.

For example, the application enable users to share, invite and plan trips with their Facebook friends, and also to retrieve booking information and see who of their friends is going to be in the same city, whether living or visiting.

To achieve that, the application was developed in Python using Django Web framework and calling the Amadeus Web Services to retrieve the flight information such as fares and bookings.

Keywords:

Facebook, trip, travel, social media, python, django

Table of Contents

1. Introduction

Since the beginning of times, human beings have lived in communities and have been influenced by their social behavior. Nowadays, with the advent of social networks, people are gradually doing more of their normal tasks in sites like Facebook and Twitter – from buying clothes to checking the traffic.

Based on that, Amadeus decided to develop a Facebook application to enable users to retrieve their bookings, share them with friends and plan trips together. This way, the experience of planning a trip and inviting friends is enhanced by the integration with the social network.

The main purpose of this Facebook application is to enable users to retrieve and import bookings from Amadeus databases using their web services with a record locator. Also, these bookings can be used to build a travel plan together (building similar trip or linking bookings) or be added to existing trips.

In the other hand, users can create new trips without a record locator by selecting options, such as departure and destination cities, class and dates, therefore saving a trip which is not booked yet.

2. Amadeus Services Ltd

Amadeus is a leading transaction processor for the global travel and tourism industry, providing transaction processing power and technology solutions to both travel providers (including full service carriers and low-cost airlines, hotels, rail operators, cruise and ferry operators, car rental companies and tour operators) and travel agencies (both online and offline).

The company acts both as a worldwide network connecting travel providers and travel agencies through a highly effective processing platform for the distribution of travel products and services (through our Distribution business), and as a provider of a comprehensive portfolio of IT solutions which automate certain mission-critical business processes, such as reservations, inventory management and operations for travel providers (through our IT solutions business).

3. Planning

3.1 Motivation

Amongst the current technology trends, one of the strongest is social media, thus emphasizing the importance of concepts such as sharing information with your friends.

In the daily life, users want to share pictures of their dinners and tell their friends that they went to the movies. Despite all of that, there are no applications that enable users to do such thing with their trips, especially those already booked in the system.

For that reason, the Amadeus We Go Facebook application is focused in that area, providing users the capability of sharing, inviting and managing trips with their Facebook friends.

In order to obtain a fully integrated experience, it was decided also that the application would be a Canvas App, which is an application inside an iFrame in Facebook.

3.2 Gannt chart of the Work Plan

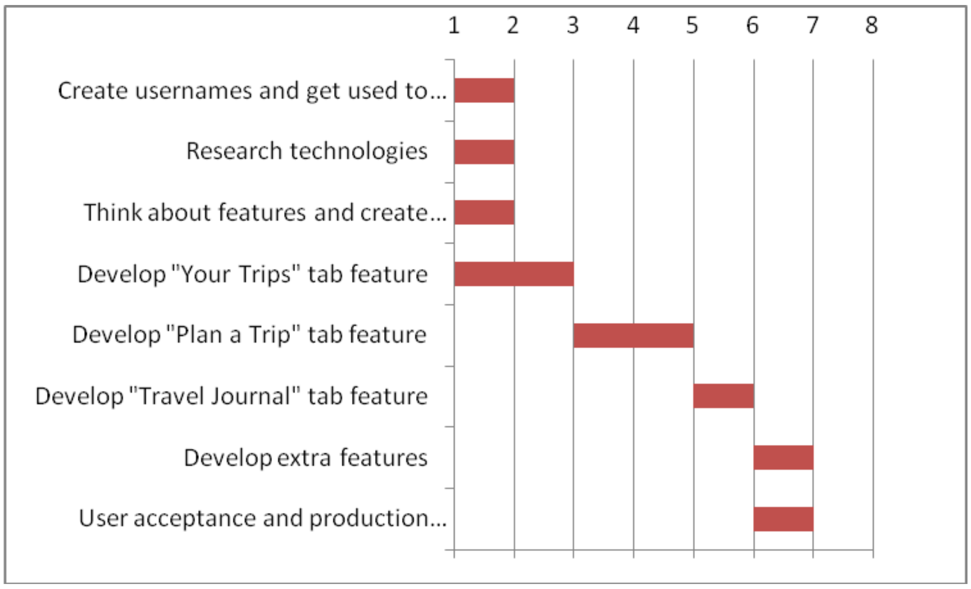


Image 1 – Gannt chart of the work plan. Y-axis is in months

3.3 Implementation decisions

In order to develop the application, two programming languages were considered: PHP and Python. While PHP has an officially supported SDK by Facebook, Python is widely used in web applications in Amadeus.

Additionally, Django Web Framework ensures many security aspects are respected and provides features such as reusability and modularity for web development with Python. For all these reasons, it was decided to use Python with Django.

Regarding the features to be implemented, several travel applications were consulted, for example Expedia and Opodo. After some research, an initial specification document was created as a reference of what was intended to be developed.

4. The Application

4.1 Features

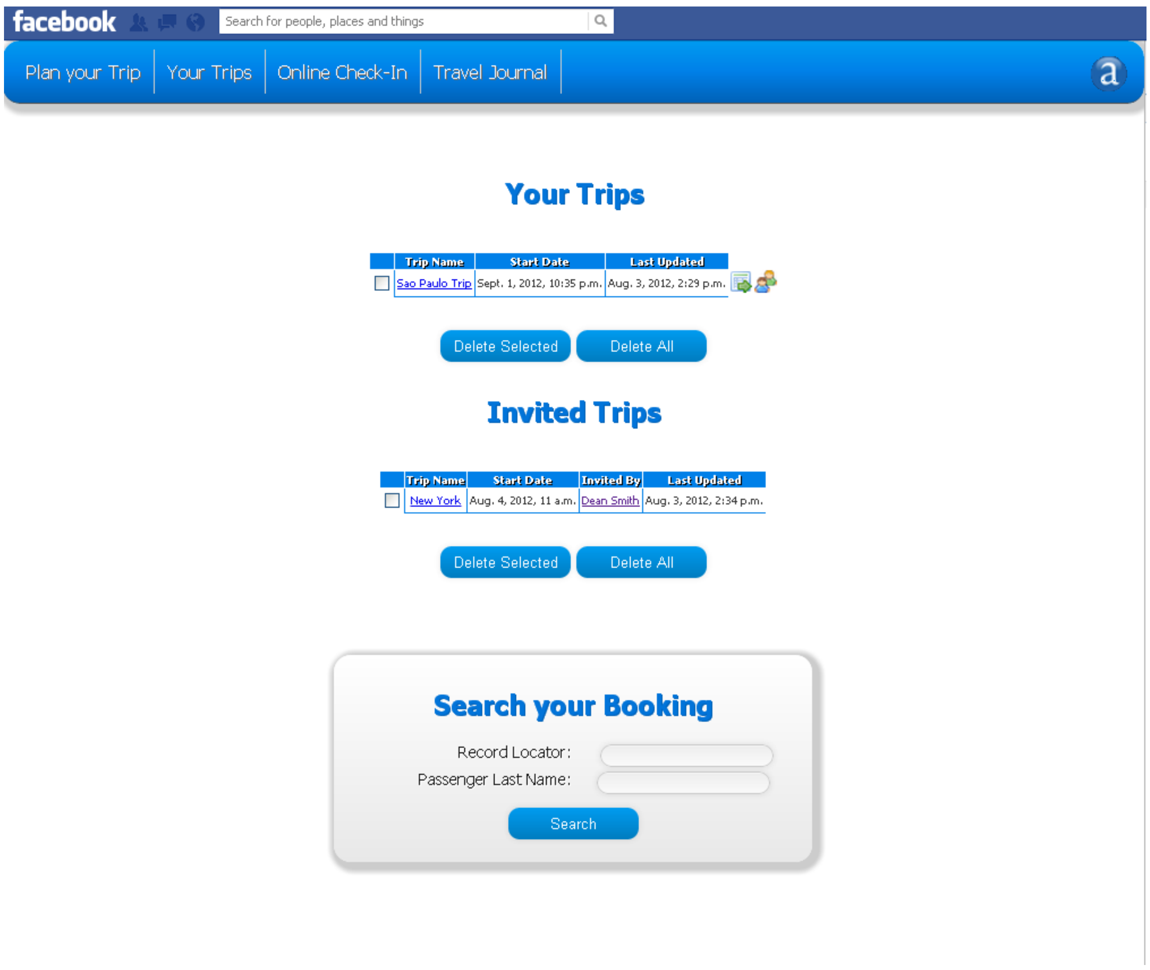
In this section, the application features will be presented and explained briefly, grouped by the context of the action performed during execution.

4.1.1 Managing Trips

In the Trips section of the application, all the user trips saved in the local database are displayed, those that are owned and those that are invitations from other users. In this screen, trips can be renamed or deleted. Clicking in the name of one of them opens the detailed information regarding that trip, such as departure and arrival city, departure and arrival date, company, flight number and others.

Also, in the right-hand side of the screen an information bar is shown displaying if the trip is linked to other trip, if there is any friend living in the destination city and if there is any friend going to the same city in similar dates. The information about linked trips is obtained from the database and the information about friends is obtained using data from Facebook and from the local database using the Facebook user id.

More specifically, if the clicked trip was an invited trip, for each flight of a trip a button is shown with the option to check availability for that specific flight. This feature will be more detailed in the chapter 4.1.3.



Your Trips

	Trip Name	Start Date	Last Updated
☐	Sao Paulo Trip	Sept. 1, 2012, 10:35 p.m.	Aug. 3, 2012, 2:29 p.m.

Invited Trips

	Trip Name	Start Date	Invited By	Last Updated
☐	New York	Aug. 4, 2012, 11 a.m.	Dean Smith	Aug. 3, 2012, 2:34 p.m.

Search your Booking

Record Locator:

Passenger Last Name:

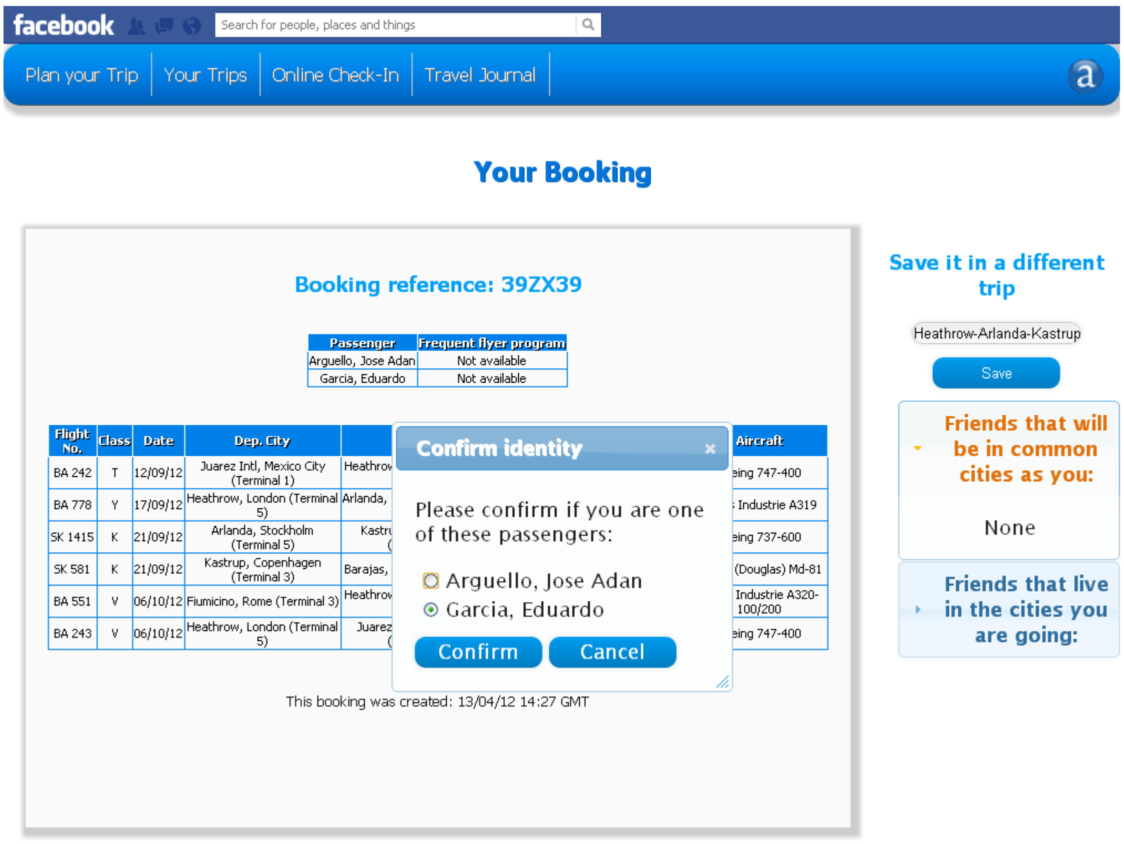
Image 2 – Application screen showing the saved trips

4.1.2 Retrieving Booking

In order to retrieve a booking, it is necessary to have a record locator, an alphanumeric 6-digit code that points to a specific record, and a surname. If the record locator exists and the surname is the same as in the record, the booking information is displayed.

All the information displayed is obtained live from Amadeus Web Services, so it is up-to-date whenever it is called. The information consists in many fields such as departure city, departure date, arrival city, airline Company, flight number, etc.

If the user does not have the booking saved in the database, a message is displayed asking if the user is one of the passengers of the booking. If so, the booking is saved in the database with the user id.



Your Booking

Booking reference: 39ZX39

Passenger	Frequent flyer program
Arguello, Jose Adan	Not available
Garcia, Eduardo	Not available

Flight No.	Class	Date	Dep. City		Aircraft
BA 242	T	12/09/12	Juarez Intl, Mexico City (Terminal 1)	Heathrov	eing 747-400
BA 778	Y	17/09/12	Heathrow, London (Terminal 5)	Arlanda,	: Industrie A319
SK 1415	K	21/09/12	Arlanda, Stockholm (Terminal 5)	Kastru (	eing 737-600
SK 581	K	21/09/12	Kastrup, Copenhagen (Terminal 3)	Barajas,	(Douglas) Md-81
BA 551	V	06/10/12	Fiumicino, Rome (Terminal 3)	Heathrov	Industrie A320-100/200
BA 243	V	06/10/12	Heathrow, London (Terminal 5)	Juarez (	eing 747-400

Confirm identity

Please confirm if you are one of these passengers:

Arguello, Jose Adan
[✓ Garcia, Eduardo]

This booking was created: 13/04/12 14:27 GMT

Save it in a different trip

Heathrow-Arlanda-Kastrup

Friends that will be in common cities as you:

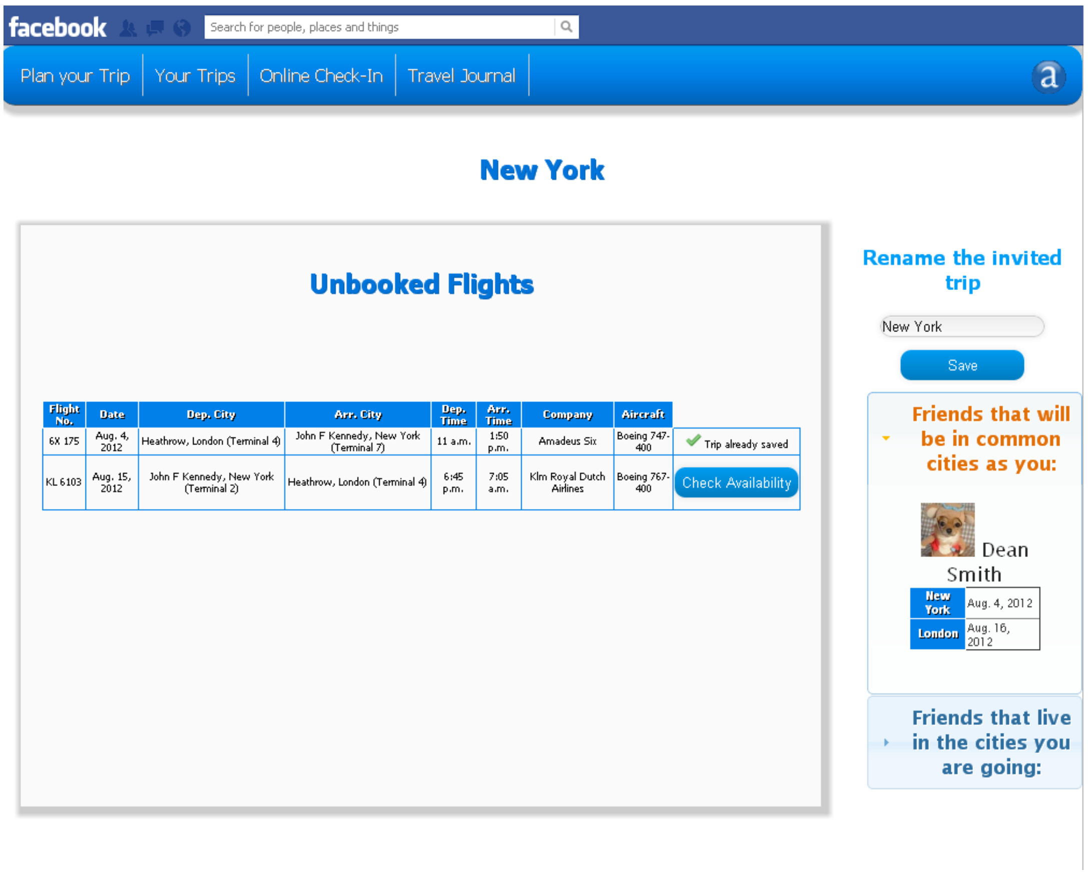
None

Friends that live in the cities you are going:

Image 3 – Screen showing a booking, confirming if user is a passenger

4.1.3 Checking an invited trip

As commented in chapter 4.1.1, the display of an invited trip includes a button to check availability for each flight in a trip. When clicked, a new screen is shown with all available flights for the same route in the same day, highlighting the flight chosen by the inviting user.



New York

Unbooked Flights

Flight No.	Date	Dep. City	Arr. City	Dep. Time	Arr. Time	Company	Aircraft	
6X 175	Aug. 4, 2012	Heathrow, London (Terminal 4)	John F Kennedy, New York (Terminal 7)	11 a.m.	1:50 p.m.	Amadeus Six	Boeing 747-400	✓ Trip already saved
KL 6103	Aug. 15, 2012	John F Kennedy, New York (Terminal 2)	Heathrow, London (Terminal 4)	6:45 p.m.	7:05 a.m.	Klm Royal Dutch Airlines	Boeing 767-400	Check Availability

Rename the invited trip

New York

Friends that will be in common cities as you:

Dean Smith

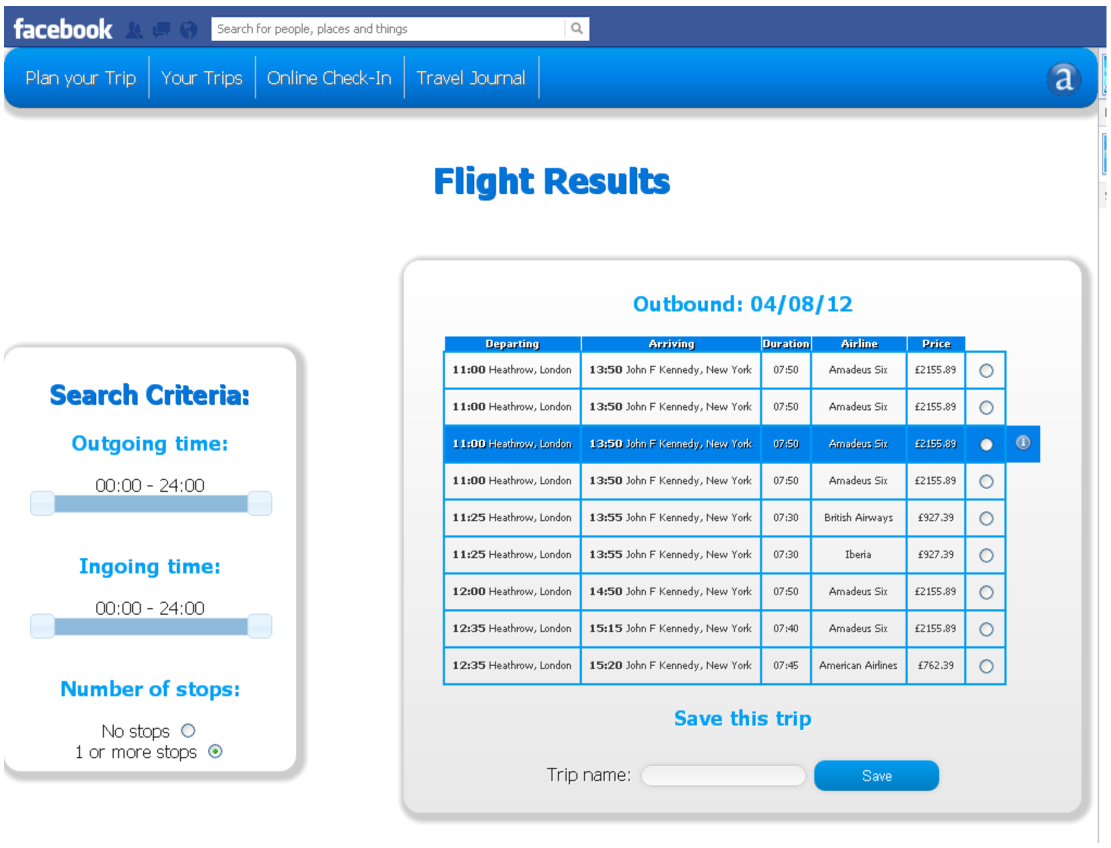
New York	Aug. 4, 2012
London	Aug. 16, 2012

Friends that live in the cities you are going:

Image 4 – Screen showing an invited trip

This gives the possibility of using an invited trip as a template, so that the invited user can select and save only the flights convenient to him/her, for example if such user is already in the second city of the route.

When saving each flight in this mode, they are all saved in the same trip, linked to the invited one in the database.



Flight Results

Search Criteria:

Outgoing time: 00:00 - 24:00

Ingoing time: 00:00 - 24:00

Number of stops: No stops ○ [✓ 1 or more stops]

Outbound: 04/08/12

Departing	Arriving	Duration	Airline	Price	
11:00 Heathrow, London	**13:50** John F Kennedy, New York	07:50	Amadeus Six	£2155.89	○
11:00 Heathrow, London	**13:50** John F Kennedy, New York	07:50	Amadeus Six	£2155.89	○
11:00 Heathrow, London	**13:50** John F Kennedy, New York	07:50	Amadeus Six	£2155.89	●
11:00 Heathrow, London	**13:50** John F Kennedy, New York	07:50	Amadeus Six	£2155.89	○
11:25 Heathrow, London	**13:55** John F Kennedy, New York	07:30	British Airways	£927.39	○
11:25 Heathrow, London	**13:55** John F Kennedy, New York	07:30	Iberia	£927.39	○
12:00 Heathrow, London	**14:50** John F Kennedy, New York	07:50	Amadeus Six	£2155.89	○
12:35 Heathrow, London	**15:15** John F Kennedy, New York	07:40	Amadeus Six	£2155.89	○
12:35 Heathrow, London	**15:20** John F Kennedy, New York	07:45	American Airlines	£762.39	○

Save this trip

Trip name:

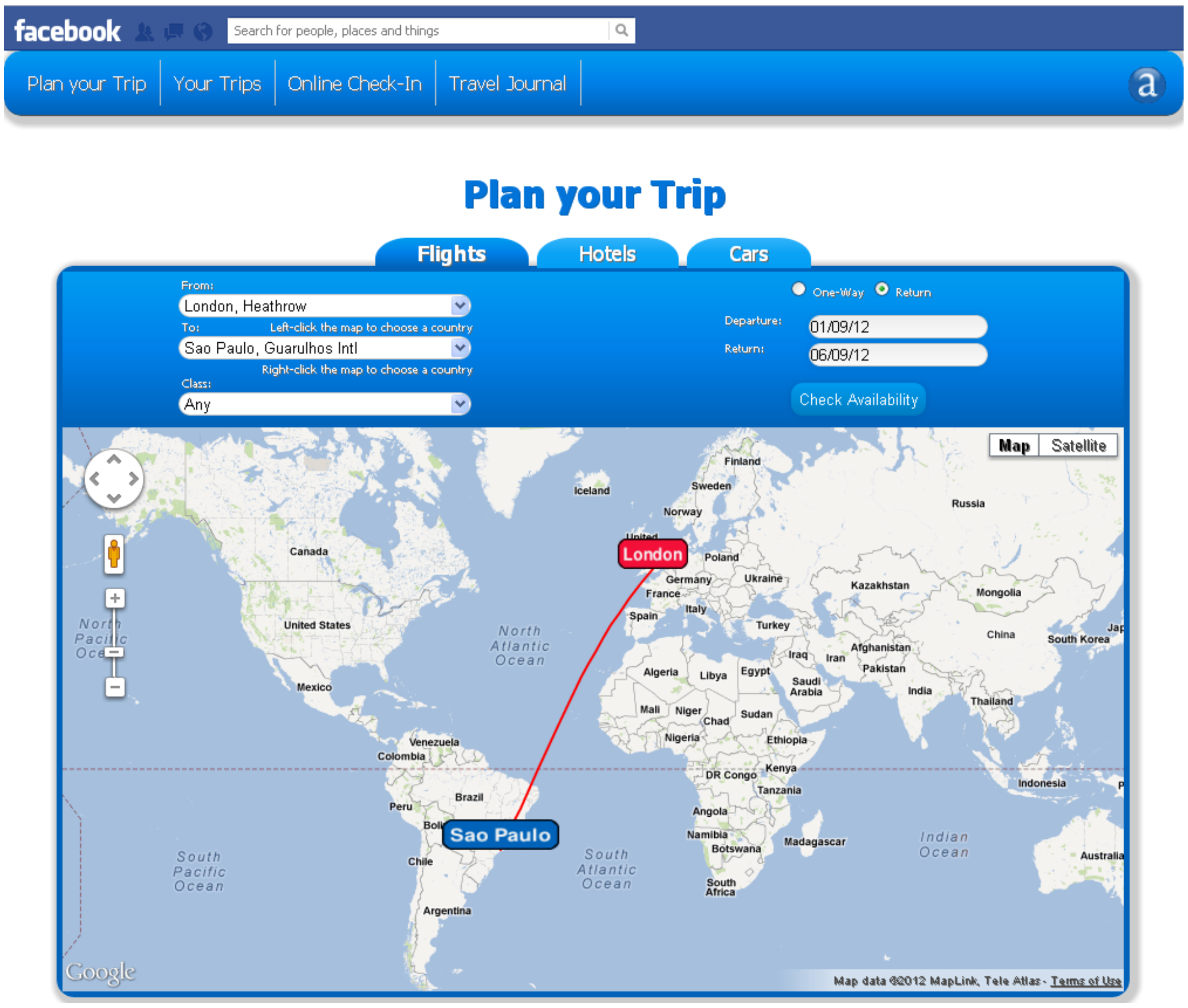
Image 5 – Screen showing the availability related to an invited trip

4.1.4 Searching available flights

In the Plan Your Trip section of the application, a screen is shown with options to search available flights, such as departure city, arrival city, class and date. Also, a Google Maps is displayed so that the user can left-click (or right-click) in a country to select as origin (or destination) country.

When a country is chosen, the city list of that option is updated with all cities from that country. This list is obtained from the Amadeus Database, which is up-to-date because it is refreshed every day.

In addition to that, clicking in countries or selecting cities creates markers in the map, providing useful information such as number of airports for the country and full name for the city.



From: London, Heathrow
To: Left-click the map to choose a country
Sao Paulo, Guarulhos Intl
Right-click the map to choose a country
Class: Any
One-Way [✓ Return]
Departure: 01/09/12
Return: 06/09/12

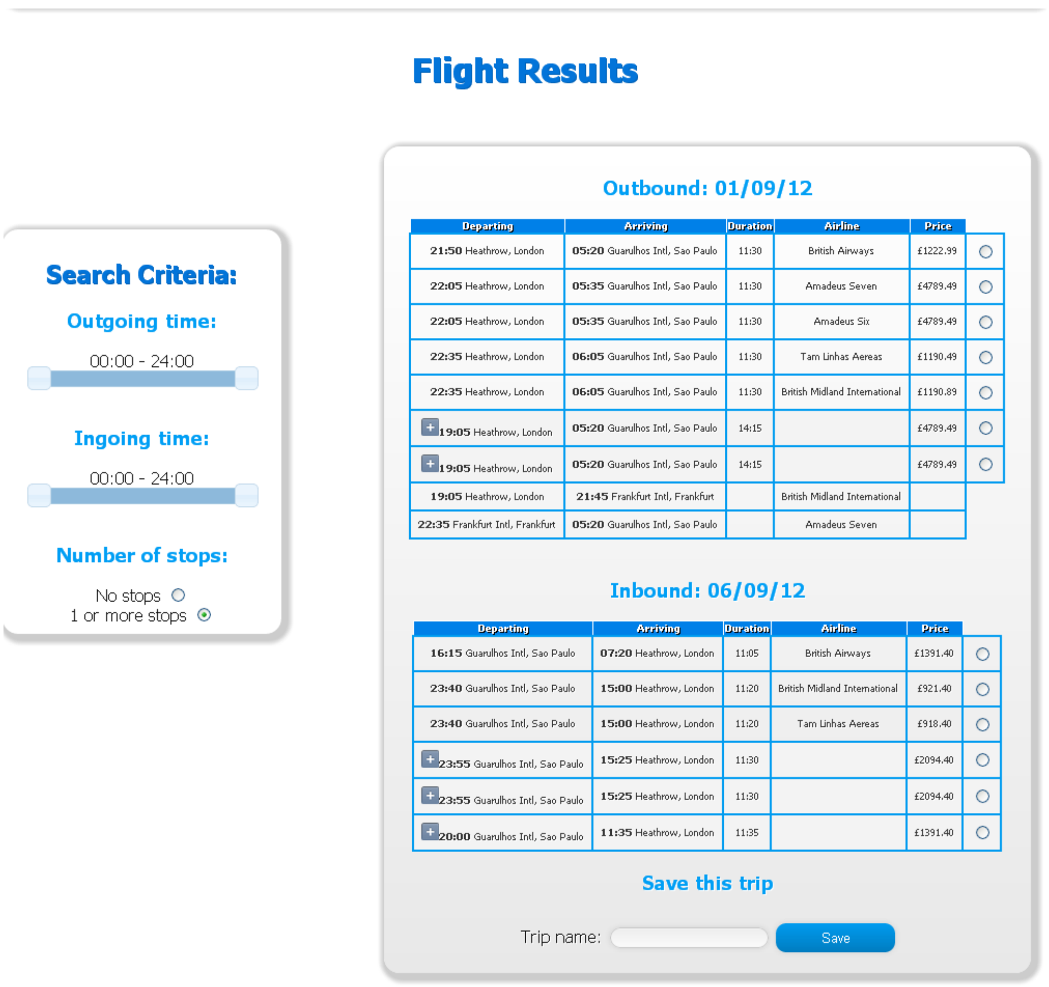
Image 6 – Screen for search of available flights

4.1.5 *Selecting available flights*

After filling the parameters to search, all the available flights are shown with information such as airline company, stops and fares, all provided by Amadeus Web Services.

In the left-hand side, a parameter bar can filter the displayed flights by departure times and number of stops. Also, clicking in a flight with one or more stops opens all the information regarding all the stops.

Finally, to save the flights in the database, the user must click in the trips and choose a name for the trip. After that, the trip is saved in the database with that name, the user id and all the important information.


Flight Results

Search Criteria:

Outgoing time: 00:00 - 24:00

Ingoing time: 00:00 - 24:00

Number of stops:

No stops ○
1 or more stops ●

Outbound: 01/09/12

Departing	Arriving	Duration	Airline	Price	
21:50 Heathrow, London	**05:20** Guarulhos Intl, Sao Paulo	11:30	British Airways	£1222.99	○
22:05 Heathrow, London	**05:35** Guarulhos Intl, Sao Paulo	11:30	Amadeus Seven	£4789.49	○
22:05 Heathrow, London	**05:35** Guarulhos Intl, Sao Paulo	11:30	Amadeus Six	£4789.49	○
22:35 Heathrow, London	**06:05** Guarulhos Intl, Sao Paulo	11:30	Tam Linhas Aereas	£1190.49	○
22:35 Heathrow, London	**06:05** Guarulhos Intl, Sao Paulo	11:30	British Midland International	£1190.89	○
+ **19:05** Heathrow, London	**05:20** Guarulhos Intl, Sao Paulo	14:15		£4789.49	○
+ **19:05** Heathrow, London	**05:20** Guarulhos Intl, Sao Paulo	14:15		£4789.49	○
19:05 Heathrow, London	**21:45** Frankfurt Intl, Frankfurt		British Midland International		
22:35 Frankfurt Intl, Frankfurt	**05:20** Guarulhos Intl, Sao Paulo		Amadeus Seven		

Inbound: 06/09/12

Departing	Arriving	Duration	Airline	Price	
16:15 Guarulhos Intl, Sao Paulo	**07:20** Heathrow, London	11:05	British Airways	£1391.40	○
23:40 Guarulhos Intl, Sao Paulo	**15:00** Heathrow, London	11:20	British Midland International	£921.40	○
23:40 Guarulhos Intl, Sao Paulo	**15:00** Heathrow, London	11:20	Tam Linhas Aereas	£918.40	○
+ **23:55** Guarulhos Intl, Sao Paulo	**15:25** Heathrow, London	11:30		£2094.40	○
+ **23:55** Guarulhos Intl, Sao Paulo	**15:25** Heathrow, London	11:30		£2094.40	○
+ **20:00** Guarulhos Intl, Sao Paulo	**11:35** Heathrow, London	11:35		£1391.40	○

Save this trip

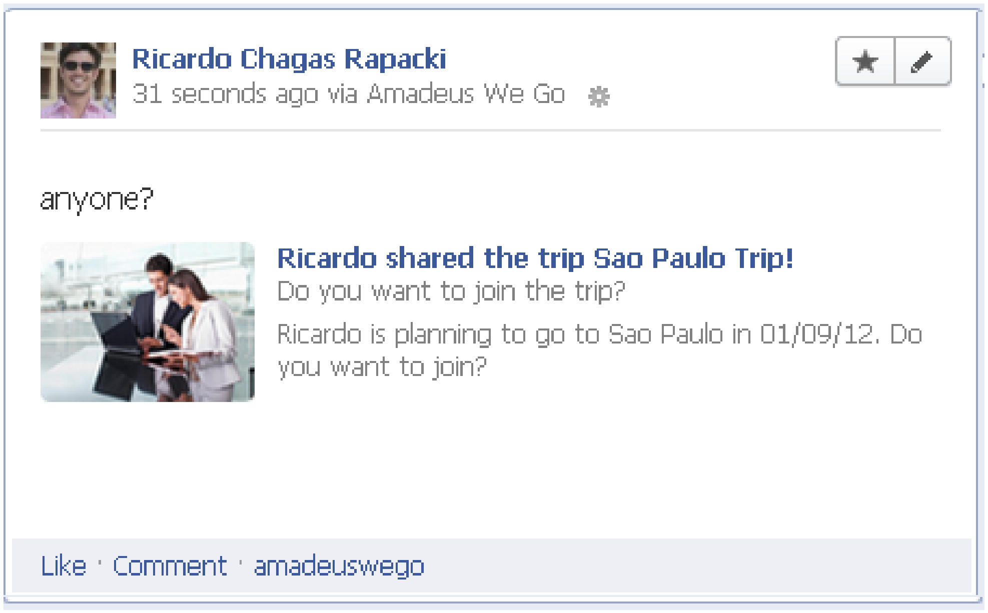
Trip name:

Image 7- Screen showing available flights for a search

4.1.6 *Posting to the Facebook Wall*

In order to post a trip to the wall, the user must click in the icon just next to the chosen trip. After that, a Facebook dialog window will appear and a message can be written to be shown in the Wall.

If other users click in the link of the post, they are redirected to the display of the invited trip, but it is not saved yet. Then, the user can save the trip as an invited trip and check availability for each flight.


Ricardo Chagas Rapacki
31 seconds ago via Amadeus We Go

anyone?

Ricardo shared the trip Sao Paulo Trip!
Do you want to join the trip?
Ricardo is planning to go to Sao Paulo in 01/09/12. Do you want to join?

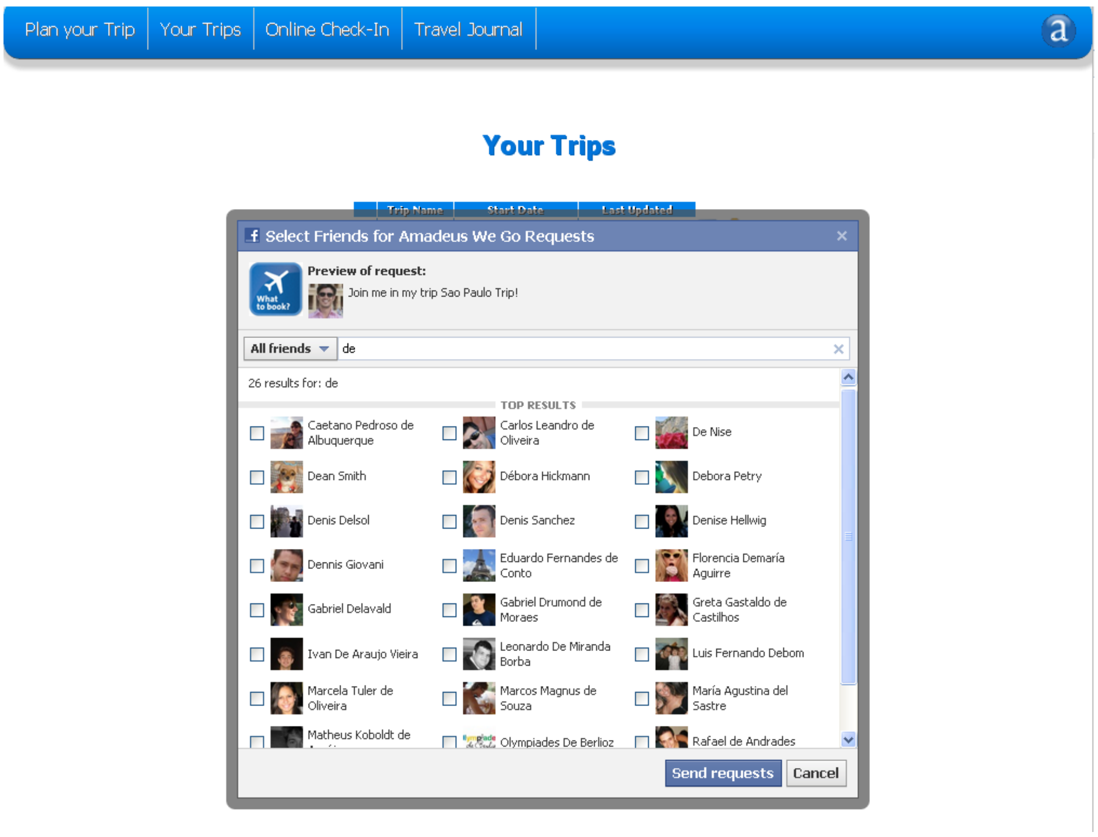
Image 8 – Screen a post of a trip to the Facebook Wall

4.1.7 Inviting a friend

To invite a friend to a trip, the user must click in the icon just next to the chosen trip and select one or more friends to invite in the Facebook dialog.

This process is made using App Requests from Facebook API via Javascript, so that the API handles all the logic of the invitation, receiving only basic parameters in the function call such as title and message.

After that, the invited user receives a notification in his/her Facebook account about the invitation and, when the application is accessed, all the invited trips are saved in the database and shown in the section *Invited Trips* to be used in the future as a template, as described in chapter 4.1.3.



Your Trips

Select Friends for Amadeus We Go Requests

Preview of request:

Join me in my trip Sao Paulo Trip!

de

26 results for: de

TOP RESULTS

Caetano Pedroso de Albuquerque	Carlos Leandro de Oliveira	De Nise
Dean Smith	Débora Hickmann	Debora Petry
Denis Delsol	Denis Sanchez	Denise Hellwig
Dennis Giovani	Eduardo Fernandes de Conto	Florencia Demaría Aguirre
Gabriel Delavald	Gabriel Drumond de Moraes	Greta Gastaldo de Castilhos
Ivan De Araujo Vieira	Leonardo De Miranda Borba	Luis Fernando Debom
Marcela Tuler de Oliveira	Marcos Magnus de Souza	María Agustina del Sastre
Matheus Koboldt de	Olympiades De Berlioz	Rafael de Andrades

Image 9 – Screen showing the invitation to a trip

4.2 Architecture

In this chapter, the application architecture and all the modules will be explained, describing the connections and the importance of each of them.

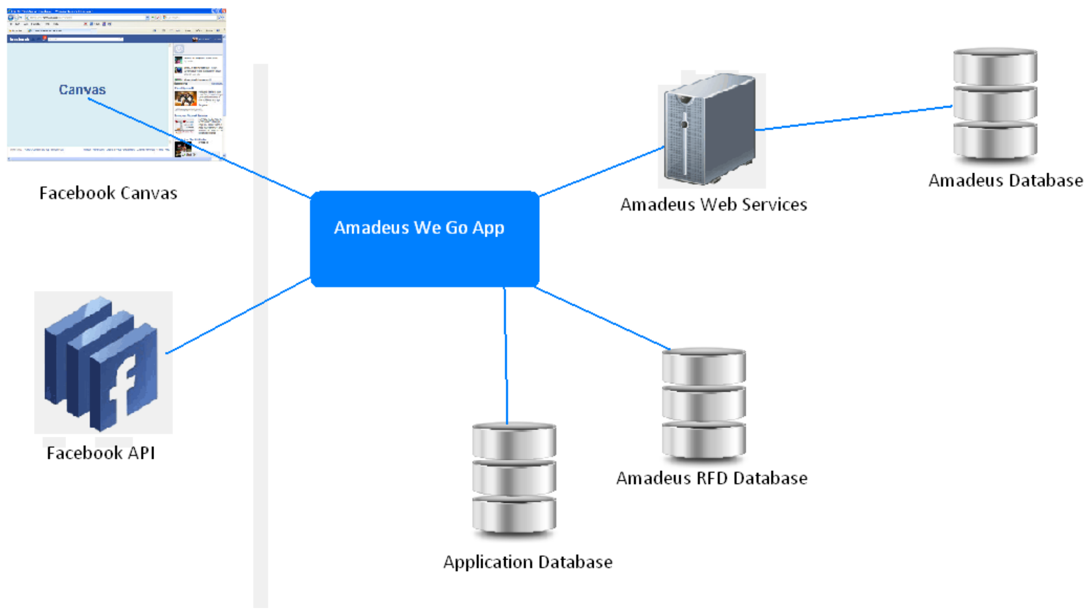


Image 10 – Application architecture

4.2.1 Facebook Canvas

The Facebook Canvas is a blank canvas inside the Facebook page where web apps are loaded, hence without leaving the page and being fully integrated with the social network.

When configuring the Facebook App in the Facebook Developers page, the application administrator must fill an URL for the Canvas Page so that the application can have a path to retrieve all HTML, Javascript and CSS files.

For example, while the address of the Canvas Application was https://apps.facebook.com/amadeuswego, the URL provided for the Canvas app in its configurations was https://lonlnx41.lon.amadeus.net.

4.2.2 Facebook API

As a Facebook application, the Facebook API is essential to use all the features and data from the social network. Since the language used was Python and Facebook does not provide directly an API for it, this module represents the open-source API called Python for Facebook, which is further detailed in chapter 5.4.1.

4.2.3 Amadeus We Go Application

This is the main module, controlling all the logic of the application and the other modules. It consists basically in a Django application, which means written in Python and using the Django web framework. In addition to that, this module includes all the HTML, Javascript and CSS files, essentials to the rendering of the web pages.

The application uses MVC (Model-View-Controller) architecture, using Django features to perform that separation: the views.py file controls which data is presented, the Django templates inside the HTML files control how the data is presented, the controller is the framework itself and the data is stored in a local database, created by Django using the file models.py.

4.2.4 Application Database

The application database is a local SQLite database automatically created by Django via the file models.py. As defined in the Django documentation website:

"A model is the single, definitive source of data about your data. It contains the essential fields and behaviors of the data you're storing. Generally, each model maps to a single database table.

The basics:

- *Each model is a Python class that subclasses django.db.models.Model.*
- *Each attribute of the model represents a database field.*
- *With all of this, Django gives you an automatically-generated database-access API. "*

In the context of the application, it is used to store information about users, saved trips and links between them. It provides an easy to use and extendable way to manipulate the stored data without the need to create directly the SQL database, tables and etc.

4.2.5 Amadeus RFD Database

The Amadeus RFD Database is stored in the company server and provides up-to-date information regarding airports, cities, airlines, terminals, countries and everything related to the travel and tourism industry.

It is updated every day and saved in a new SQLite file, therefore every first execution of the program runs a script to check the newest database in the folder to use it.

Examples of use of the database include retrieving the name of the city and airport from the 3-digit code, such as CDG to Paris Charles de Gaulle, and listing all airports from a specific country.

4.2.6 Amadeus Web Services and Database

Responsible for the integration between the application and the services provided by Amadeus, this module is called by the application to provide information stored in the company database via SOAP requests and returns a SOAP message with all useful information for that service.

More about this module is explained in chapter 5.4.2, detailing how the requests are done by the application and how it parses the response.

4.3 Deployment

Initially, for the deployment of the application, the Django development server was used, a lightweight Web server written purely in Python and included with Django. However, this decision resulted in a problem with Facebook because the server does not support native HTTPS, which is mandatory by Facebook for Canvas Applications.

For that reason, an Apache Server was configured in the Linux development box, providing HTTPS support thus enabling Facebook Canvas to fetch the page from the URL.

4.4 Technologies

In this chapter, the technologies already presented in previous chapters will be described in detail to fully comprehend how they are used by the application and what is their importance.

In order to satisfy all requirements and provide a good user experience, the Amadeus We Go Facebook application was developed using many languages and frameworks: Python with Django, Javascript with jQuery, HTML and CSS. The graphic above details the number of lines written and the percentage for each language.

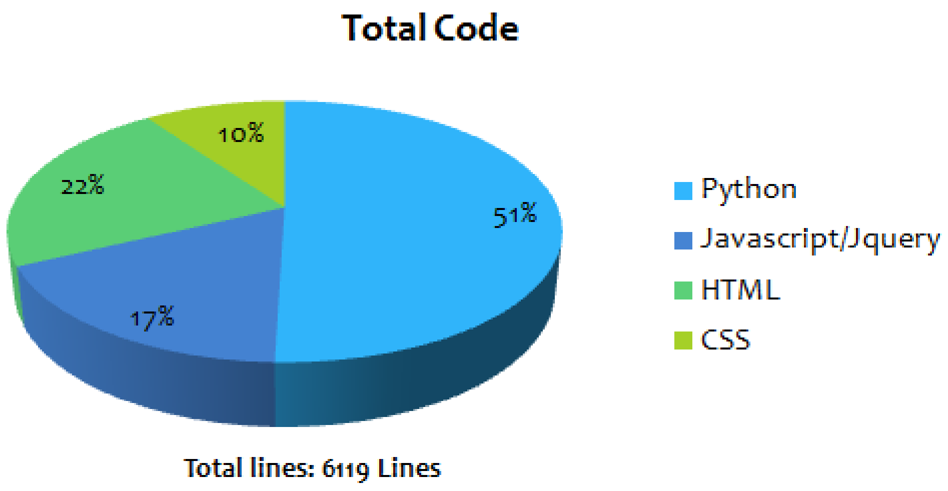


Image 11 – Number and percentage of lines written in the application

4.4.1 *Facebook API*

As a Facebook Application, it is of natural importance to use the Facebook API. Facebook officially provides nowadays SDKS for Javascript, PHP, iOs and Android. As for the other programming languages, it is necessary to perform HTTP requests and parse the JSON response.

4.4.1.1 *Python for Facebook*

For the Amadeus We Go Application, it was decided to use the open-source SDK called Python for Facebook to abstract those operations and provide easy access to actions and data from the social network. One of the most important reasons for that choice was the portability of the SDK and the large community that supports it.

The code is stored in Github (http://github.com/pythonforfacebook/facebook-sdk) and PyPi (http://pypi.python.org/pypi/facebook-sdk) and to use it is necessary only to copy the facebook.py file to the project directory and include it in the main python file. However, it is necessary to use Facebook Javascript SDK to implement Facebook authentication in the canonical way and, after that, parse the cookie set in the python file to start using the SDK.

4.4.1.2 OAuth 2.0

In order to authorize users in Facebook Applications, the Facebook API uses OAuth 2.0, a protocol originally created in late 2006 and, as defined in the OAuth website, "*focuses on client developer simplicity while providing specific authorization flows for web application and others*".

One of the main advantages of this protocol is the fact that the application does not obtain the username and password because the authorization flow is controlled by Facebook itself. The application requests an authorization to the user, the user is redirected to a Facebook dialog to send username and password, and if correct Facebook returns a user access token.

When requesting an authorization, the application must specify what are the permissions needed in Facebook, for example accessing pictures or posting to the Facebook Wall. These permissions can be revocable or not and are shown to the user in the Facebook dialog. Using OAuth 2.0, the access token generated in the end of the process contains only those permissions. Every time new permissions are needed, a new dialog is shown to create a new access token.

The permissions requested by the Amadeus We Go Facebook application are user_likes (for the liked pages of the user), friends_about_me (for personal info about the user's friends), friends_location (for the city of residence of the user's friends) and publish_stream (to publish posts to the Facebook Wall). The last one is the only one which is individually user-revocable; the others are mandatory to use the application.

4.4.1.3 Graph API

Nowadays, the core of the Facebook API is the Graph API, which, as defined in the Graph API website, "*represents a simple, consistent view of the Facebook social graph, uniformly representing object in the graph (e.g., people, photos, events, and pages) and the connections between them (e.g., friend relationships, shared content, and photo tags)*".

So, using Python for Facebook, after retrieving the object Graph, anything in the social network can be accessed by its ID and connections. For example, after accessing a user by its Facebook id, all friends, liked pages and pictures can be easily retrieved using the connections between them.

4.4.2 Amadeus Web Service

As explained in chapter 4.2.6, the Amadeus Web Service is the responsible to retrieve up-to-date information from the company database and provide data regarding flight availability, fares and booking information.

To use it, a SOAP Client must send SOAP requests to the web service and parse the SOAP response to retrieve the information. In this application, the python file client.py implements it and stores all the necessary information to call the web service: the service URL, session id, security token and sequence number.

Each call to the Web Service must include the session id, security token and sequence. In the first call, the session id and security token are empty and are returned in the SOAP header of the response. The sequence number must start at 1 and be increased by the SOAP client each call to the web service so errors can be detected by analyzing the header.

While that, all the information regarding the SOAP body are included in other python files, one for each service used. They contain a template of the SOAP message, the parameters passed for the message and the logic to parse the response, generating a dictionary of dictionaries and lists.

In the next chapters, each service used will be briefly explained.

4.4.2.1 Security Authenticate

This is the only mandatory service to any application that uses the Web Service and it is always the first one to be called because it performs the authentication on Amadeus application, secured by the Logon and Security Server (LSS), a unique identifier for each application or employee.

The parameters used by this service are username, base64 encrypted password, length of original password, office code, duty code and organization code. The response includes the status code, the session id and security token, necessary to perform further operations.

4.4.2.2 PNR Retrieve by Record Locator

This service is responsible for retrieving information from an active passage name record (PNR) using a record locator, a 6-digit alphanumeric code.

The only parameter used by this service is the record locator of a booking, since it is a unique code. The response includes a large range of information regarding the booking, for example details about passengers, departure and arrival location, dates, company and aircraft.

4.4.2.3 Flight Availability

This service provides availability information for flights on more than 360 airlines, such as Lufthansa, Airfrance, Brittish Airways, TAM and more.

The parameters used by this service are departure airport code, departure date and time, cabin option and arrival airport code, where the airport codes are 3-digit IATA codes. Cabin option is a number identifying different groups of classes, the most important being first class, business class and economy class. The response includes information about all available flights, such as departure time in the chosen date, airline, segments of the flights and airline. Also, the number of seats available per class is retrieved.

4.4.2.4 Itinerary Fare

This service provides fares for passenger types without existing reservation. In other words, up-to-date fares if the passenger wanted to know the price to buy a ticket at the moment of the search.

The parameters of this service consist of the currency to show the fares, departure date and airport, arrival city, airline and flight number. The response includes the amount in the chosen currency for each type qualifier, where only the full price (including taxes) is used in the application.

4.4.3 Other technologies

Amongst the other technologies used, there are APIs and libraries to enhance the user experience adding more information and features. One of them is the jQuery, a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development.

This library is used in Javascript files for a better and cross-browser web development and also for its UI, jQuery UI, which provides abstractions for low-level interaction and animation, advanced effects and high-level, themeable widgets, built on top of the jQuery JavaScript Library.

In addition to that, Google Maps Javascript API was used, in order to achieve a better user interface by providing a Google Maps map in the application so the user can select countries and cities to travel from and to.

All of them were very simple to use and quick to integrate to the application, helping the interaction between the application and the user and providing cross-browser portability for the application.

4.5 Database

In this application, the local database is used to store information about users, passengers, trips and links between them, avoiding the need to call the Web Service whenever trip information is needed. It also allows the application to manipulate two different types of booking: those with a record locator, already bought and confirmed, and those without it, created in the application.

In the picture above, the application database schema is shown and in the next chapters each table and their relationships will be further explained.

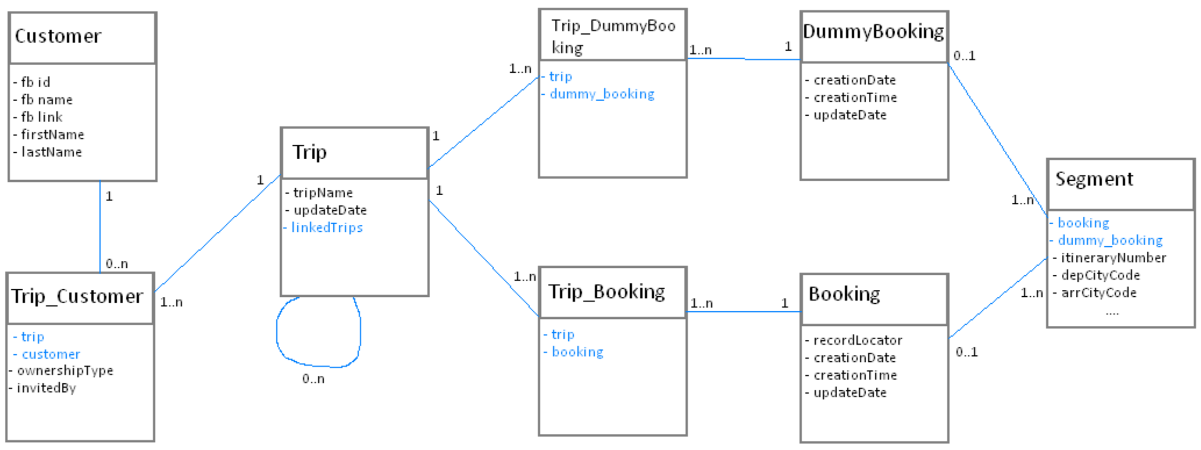


Image 12 – The Application Database Schema

4.5.1 *Customer Table*

As the name suggests, this table represents a customer, either a Facebook user or a stored passenger from a booking. For that reason, it contains two different types of information: those regarding a Facebook account and those regarding passenger information from a booking.

A customer may be created in two moments:

- during the save of a new trip, storing Facebook Id, full name and link to the Facebook page to avoid calling the Facebook API thus enhancing performance

- during the retrieve of a booking by a record locator, storing all passengers from that booking in the database without any Facebook information

In the second case, if the user confirms being one of the passengers, that customer in the database is updated with the user Facebook information.

4.5.2 Trip Table

This table represents the concept of a trip, storing useful information and the link relationship between different trips. The fields of a trip are the name, the update date and a many-to-many field pointing to its linked trips.

More specifically, the update date is the last date when the user clicked in one of the bookings belonging to the trip, pulling up-to-date information from the Web Services.

A trip is created whenever a user plans and stores a new trip using the application or a new booking is retrieved from the Web Services using a record locator.

4.5.3 Trip_Customer Table

This table represents the many-to-many relationships between trips and customers, since a customer may have many trips and a trip may belong to many users.

Its fields include a foreign key to the Trip table, a foreign key to the Customer table, an ownership type, which defines if the user is the owner of a trip or was invited, and in the last case a field to store the inviting customer. Otherwise, that field is empty.

Also, it possesses the information if a trip is owned by a customer or not. If no customer owns a specific trip, it must be deleted.

4.5.4 Booking

As commented previously in chapter 4.5, there are two types of booking, with or without record locator, and this table represents the first one. Its primary key is the record locator and other fields include creation date and time and update date.

As well as the Trip table, the update date is the last date when the booking information was pulled from the Web Services. A booking is created in the moment a user searches one by its record locator for the first time.

4.5.5 DummyBooking

This table represents a booking without a record locator, created in the application and without a matching data in the Amadeus database. It contains the same fields as the table Booking, with the exception of the record locator.

Its importance is to provide the user the capability to create new bookings as if they were confirmed trips. The creation occurs in the moment a user saves a trip after searching available flights or checking an invited trip.

4.5.6 Trip_Booking

This table represents the many-to-many relationships between trips and bookings, since a trip may have many bookings and a booking may belong to many trips.

Its fields include only a foreign key to the Trip table and a foreign key to the Booking table. Nevertheless, a many-to-many field could be stored in the Trip table pointing to the Booking table instead, but this table was created expecting that, in the future, some extra information in the relationship could be needed.

4.5.7 Trip_DummyBooking

This table represents the many-to-many relationships between trips and dummy bookings, since a trip may have many dummy bookings and a dummy booking may belong to many trips.

Analogous to the Trip_Booking table, the only difference is that, instead of a foreign key to the Booking table, it contains a field for the foreign key to the DummyBooking table, as the name suggests.

4.5.8 Segment

This table represents a segment of a flight, containing all the useful information about trips and bookings. Besides this data, it contains a foreign key to the Booking table and the DummyBooking table, mutually exclusive (only one is not null), so that a segment may belong to either a booking or a dummy booking.

For that reason, a segment is always linked to a booking or a dummy booking, so its creation and destruction occurs in the same moment of them. The rest of the fields of this table include itinerary number, arrival and departure city, arrival and departure date, company, aircraft, fare and if the segment is the last of a booking.

4.6 Testing

In order to store the application source code, a distributed revision control called Mercurial was used to manage the repository and its operations.

Whenever a version of the application was going to be stored locally, the developer had to commit it to a repository in the local machine. After that, the developer could nominate the modification so a request to the integrators would be created to authorize the operation.

Then, an integrator would download the source code, run a new Apache server and test if all the features were functional and no errors were found. In that case, the integrator would push the modifications to the Mercurial repository so any further pulls would have the latest and functional source code.

5. Result Analysis

In this chapter, the implementation decisions made and the final result achieved in the program will be analyzed, judging what the benefits were and what the drawbacks were. Also, future improvements will be commented and finally an updated Gannt chart will be presented for what was really followed.

5.1 Benefits

During the planning, one of the most useful decisions was to choose Python as the main language and Django as framework, instead of PHP. In spite of that, the beginning of the development of the SOAP client was complicated due to that choice, because three python SOAP frameworks were tested without success: SUDS, SOAPpy and ZSI.

However, after achieving that using the Python native module urllib2, the development was only facilitated by the object-oriented language, its modules and all the support to Web-development from Django.

For example, even though the Python for Facebook SDK is not supported by Facebook, it was very simple and powerful to use and didn’t cause any problems. Also, the Django object-relational mapper called Models transforms a database layout in python code to SQL syntax, thus being more easily modifiable and organized.

5.2 Drawbacks

As commented in the previous chapter, one of the main problems encountered during the development was the creation of a SOAP client. The attempt with three different frameworks caused a delay to the development of one month, which was crucial to develop all features planned.

The cause was that, for each framework, an inconsistency was encountered with the SOAP message being sent and the one expected by the Web Services. Also, the lack of documentation regarding how to send SOAP messages to the Web Service was problematic.

Another drawback, but not so big, was the fact that an ideal database schema wasn't planned in the beginning, resulting in many changes to the Models file and the python file to retrieve data from the database.

In addition to that, sending HTTP requests with Facebook API and Web Services decreased the performance of the application. While calling the Web Service is time consuming, using the Facebook API is not so slow but may be called several times, for example to check all of the user friends information.

However, this performance issue could be partially solved if the application was deployed in the Erding server, where the Web Services are located. This was not done only because only applications in production systems are deployed in that server.

5.3 Improvements

After analyzing the benefits and drawbacks, the possible future improvements to the application are considered and commented, whereas it is a correction to a drawback or a new feature.

First, as commented in the previous chapter, it would be crucial to deploy the application in the Erding server to improve the performance hence improving the user experience. Also, solutions should be discovered to speed up the Facebook API calls and to perform fewer requests, for example caching information about the user friends, so that the Facebook API performance would be better as well.

Also, many features could be implemented and would be very useful to the user. For example, a part of the application called Travel

Journal where the user could give feedback about trip locations and hotels and create albums for the trips, so his friends could see.

To improve the sharing of a trip, a section of the application could manage a group of users, their suggestions and their votings so that decisions about future trips would be easier to be made when travelling in a group.

In addition to that, the user could be informed by the application of the documentation and visa needed when searching available flights, analyzing up-to-date information about the relationship between departure and arrival countries.

One of the main improvements and most complicated would be to allow the user to buy plane or train tickets or book hotels and not only create them in the database, so that the application could control the complete flow of a trip. However, there are many business rules applied regarding the airline companies and an extensive study should be made to discover how it should be done.

Finally, other minor features would be useful, such as allowing the user to update the fares from the Web Service to the unconfirmed flights when desired, show the total price of a trip, etc.

5.4 *Gannt chart of the development achieved*

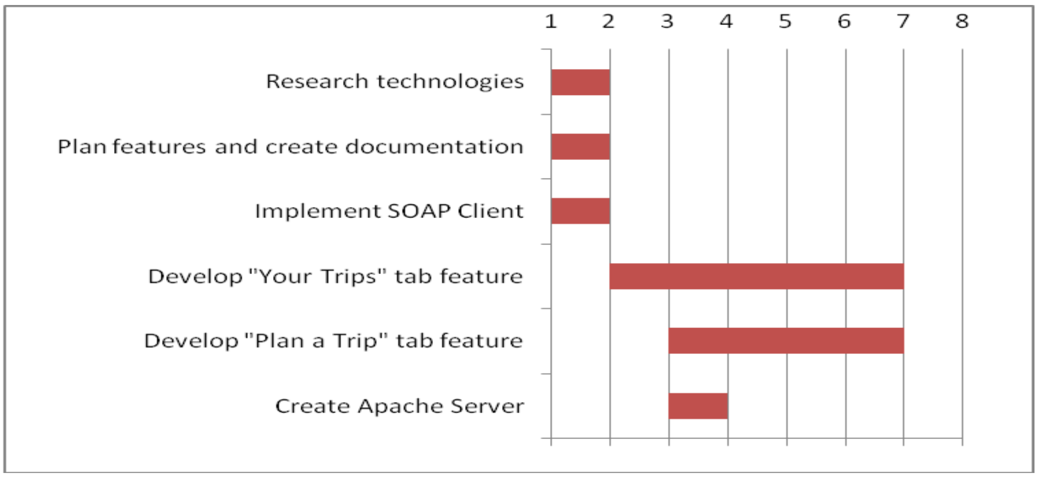


Image 13 - Gannt chart of the work plan achieved. Y-axis is in months

6. Conclusion

From this report, even though not all features planned were implemented, the main flow was achieved and the final application addresses completely the initial problems that generated the motivation.

The module that required more efforts during the development was the integration with the Amadeus system, especially how to call and parse the information from the Web Services, as noted in the Drawbacks chapter. In contrast, Python, Django and Python for Facebook SDK didn't generate any issues and were simple and powerful to use.

In order to support new features and to update the existing ones, the extension of the application is simple and offers a great capacity for the application to be expanded. These extensions require only modifying the Web Service which is called, parsing the new data and including more Facebook interaction such as posting to the Facebook Wall when creating a trip and suggesting friends to travel with based in the common likes.

To conclude, the internship was challenging and very important to learn many different technologies and integrate them with already existing production systems from Amadeus. The final application developed solves the initial problems proposed and is easily extendable, so that the project can be continued and perhaps become a live application that will provide a new way of organizing trips with friends.

Bibliography

1. **Django** - The Web framework for perfectionists with deadlines. https://www.djangoproject.com/

2. **Python for Facebook** - A home for Python developers working with Facebook. http://www.pythonforfacebook.com/

3. **Amadeus**. http://www.amadeus.com/amadeus/x5034.xml

4. **Amadeus Web Services**. http://webservices.amadeus.com/

5. *W3C (2007, April, 27) - **SOAP Version 1.2 Part 1: Messaging Framework (Second Edition).*** http://www.w3.org/TR/soap12-part1/

6. **SUDS** - a lightweight SOAP python client for consuming Web Services. https://fedorahosted.org/suds/

7. **SoaPpy** - A SOAP/XML Schema Library for Python.

http://soapy.sourceforge.net/

8. **ZSI: the Zolera Soap Infrastructure** - a Python package that provides an implementation of SOAP messaging.

http://pywebsvcs.sourceforge.net/zsi.html

9. **Google Maps Javascript API v3** - the solution for Maps Applications for both Desktop and Mobile Devices.

https://developers.google.com/maps/documentation/javascript/

10. **jQuery** - a fast and concise Javascript library that simplifies HTML traversing, event handling, animating, and Ajax interactions for rapid web development, . http://jquery.com/

11. **jQuery UI** - provides abstractions for low-level interaction and animation, advanced effects and high-level, themeable widgets. http://jqueryui.com/

12. **Opodo.** http://www.opodo.co.uk/

13. **Expedia.** http://www.expedia.co.uk/

14. **Graph API** - presents a simple, consistent view of the Facebook social graph, uniformly representing objects in the graph and the connections between them.

https://developers.facebook.com/docs/reference/api/

15. **Facebook Javascript SDK** - provides a rich set of client-side functionality for accessing Facebook's server-side API calls.

https://developers.facebook.com/docs/reference/javascript/

16. **Mercurial** - a free, distributed source control management tool. http://mercurial.selenic.com/

Grenoble INP - Ensimag
École Nationale Supérieure d'Informatique et de Mathématiques Appliquées

Rapport de projet de fin d'études

Effectué chez Amadeus Londres

Altea Booking integration into Social Networking

Ricardo Chagas Rapacki

3e année -— Option ISI

Août 2012

Amadeus Services Limited	Responsable de stage
World Business Centre Building 3	Amit Gawali
1208 Newall Road	
Middlesex, TW6 WTA	
United Kingdom	Tuteur de l'école
	Claudia Roncancio

Abstract

Intéressé par l'intégration des services fournis par l'entreprise avec des réseaux sociaux, Amadeus a décidé offrir une opportunité de stage afin de développer une application Facebook, dans lequel les utilisateurs peuvent récupérer ses bookings et les partager avec ses amis, entre des autres fonctionnalités.

La principale intention de l'application consiste en utiliser les fonctionnalités du Facebook de façon à permettre les utilisateurs à partager des voyages avec ses amis et les données stockées dans le réseau social pour fournir des informations utiles dans l'application.

Par exemple, l'application permettre les utilisateurs à partager, inviter et planifier les voyages avec ses amis sur Facebook, et aussi à récupérer information sur le booking et voir qui entre ses amis sur Facebook va être dans la même ville, soit ils vivent ou en visitant.

De façon à réaliser ça, l'application a été développée en Python en utilisant le Web Framework Django et en appelant les Web Services d'Amadeus pour récupérer des informations tells que des tarifs et informations sur des booking.

Mots-clés:

Facebook, voyage, social media, python, Django, réseau social