

ATRS

Air Ticket Reservation System

System's Architecture and Design

Summary

SUMMARY	2
SYSTEM'S ARCHITECTURE AND DESIGN	3
PURPOSE OF THIS DOCUMENT	3
SYSTEM'S OVERVIEW	3
PHYSICAL VIEW.....	5
KEY COMPONENTS OF THE SOLUTION.....	5
ARCHITECTURE OVERVIEW.....	5
ATRS-CLIENT	6
<i>Development environment.....</i>	<i>6</i>
<i>Authentication Mechanism.....</i>	<i>6</i>
<i>Client-Server Communication.....</i>	<i>7</i>
ATRS-SERVICE.....	7
ATRS-MS	8
DETAILED VIEW	9
PROGRAMMING LANGUAGES.....	9
COMMUNICATION AND DATA STORAGE.....	9
SECURITY AND AUDIT	10
DESIGN AND CODING PATTERNS.....	10
LOGICAL VIEW	11
DATA MODEL.....	13
HUMAN INTERFACE DESIGN	14
<i>Login.....</i>	<i>14</i>
<i>Sign up.....</i>	<i>15</i>
<i>Flight search.....</i>	<i>16</i>

System's Architecture and Design

Purpose of this document

The main purpose of this document is to describe the architecture and design of the ATRS Software in many ways, including the established strategic decisions about the structure and behavior, collaborations between its components, and its physical elements.

We will go deep in the parts of the solution, including functional, logical and physical elements, and every requirement that justify the fundamental mechanisms of the system, that is, basic design decisions primarily based on non-functional requirements of the product, which define design patterns and implementation shared by all team members.

System's Overview

The ATRS is a software for air ticket reservation and must be available for public customers and the airline internal staff. The main functional requirements are listed below:

1. Public users
 1. Are able to login using their google+, twitter or LinkedIn accounts.
 2. Are able to reserve air tickets searching from multiple flights based on multiple criteria.
 3. Can make payment using credit cards.
 4. Receive tickets as PDF via email. This email should contain useful links like "My Bookings", "Checkout Now", "Cancel booking" that will not require the user to login.
 5. Are able to perform online checkout/cancellation, from 48 and up to 4 hours from departure, choose seats, and print boarding passes.
 6. Receive email alert at 48 hours from departure, regarding their reservation, to encourage them for early online checkout. This email should contain a "Checkout Now" link that will not require the user to login.
2. Airline staff
 1. Are able to login using their active directory credentials, even from public computers.
 2. Receive reservation chart as PDF at 1 hrs from departure.

3. Cancel any ticket, generating email alert and refund to the public user.
4. Cancel the whole flight, generating email alerts and refunds to the public users.

In addition to the functional requirements, there are some non-functional requirements that must be fully satisfied in the project presented:

1. Must create a Single Page Application (SPA) based web portal with responsive design.
2. System core must be ready to serve mobile apps in future w/o any modification.
3. System will be installed on cloud with services that you recommend enabled.
4. All the alerts sent should be guaranteed (not lost due to some technical errors).

Physical View

Key components of the solution

The complete solution to be developed will cover a web application to be the client (atrs-client), a rest full application to be the server side (atrs-service), and a complementary application to guarantee the delivery of each messages that the application needs to send (atrs-ms).

The third application will run independently of others applications because it is highly desirable to isolate the batch process from the online process.

Architecture overview

The complete solution will be divided in three parts just as exposed in the next figure:

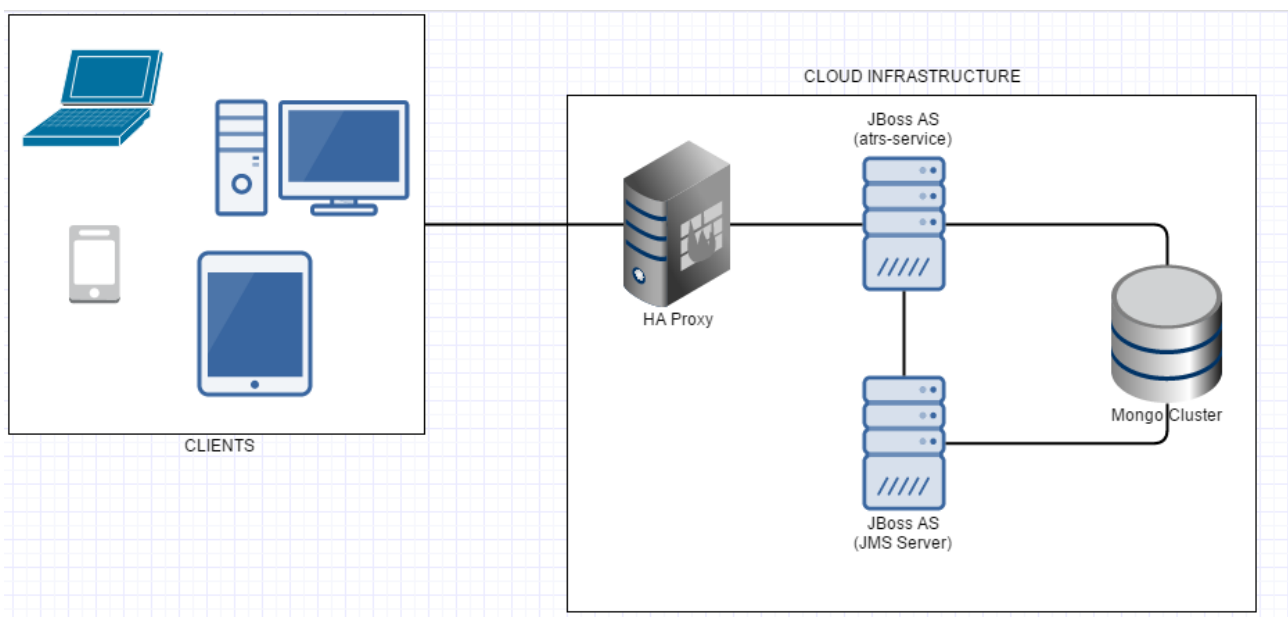


Figure 1 - Architecture overview

ATRS-client

ATRS-client is a web client developed to run in conventional web browsers and mobile devices. The application was designed to fulfill the following requirements:

- The application must run in different environments and must be responsive;
- The application must be read to serve mobile apps in future without any modification;
- The application will send emails with useful links that will not require the user to login

The application consists in a bunch of web pages developed using angular-js in conjunction with angular material components. In addition to providing a complete library of rich components and 100% responsive, the angular-material is fully compatible with almost all versions of angular-js and has interesting features for mobile and touch-screen-enabled devices.

Development environment

To facilitate the development of the interfaces with multiple visual components from different libraries, we need our application to support installation of components packages from public repositories. In addition to it, we must provide a mechanism to allow compression and minification of the resources to decrease the network payload.

Bower is a great tool to keep track of all the packages used in the view layer of the application and make sure they are up to date and fully compatible.

Gulp.js is a modern build tool used to automate tasks such bundling and minifying libraries and stylesheets, refreshing the browser where you save a file in development environment, quickly running unit tests, code analysis, etc.

Authentication Mechanism

Based on the requirements that we will have to provide a custom authentication mechanism that supports sending authentication tokens through mail, the only option is to use a token-based solution. The next figure shows a comparison between token and cookie based authentication mechanisms:

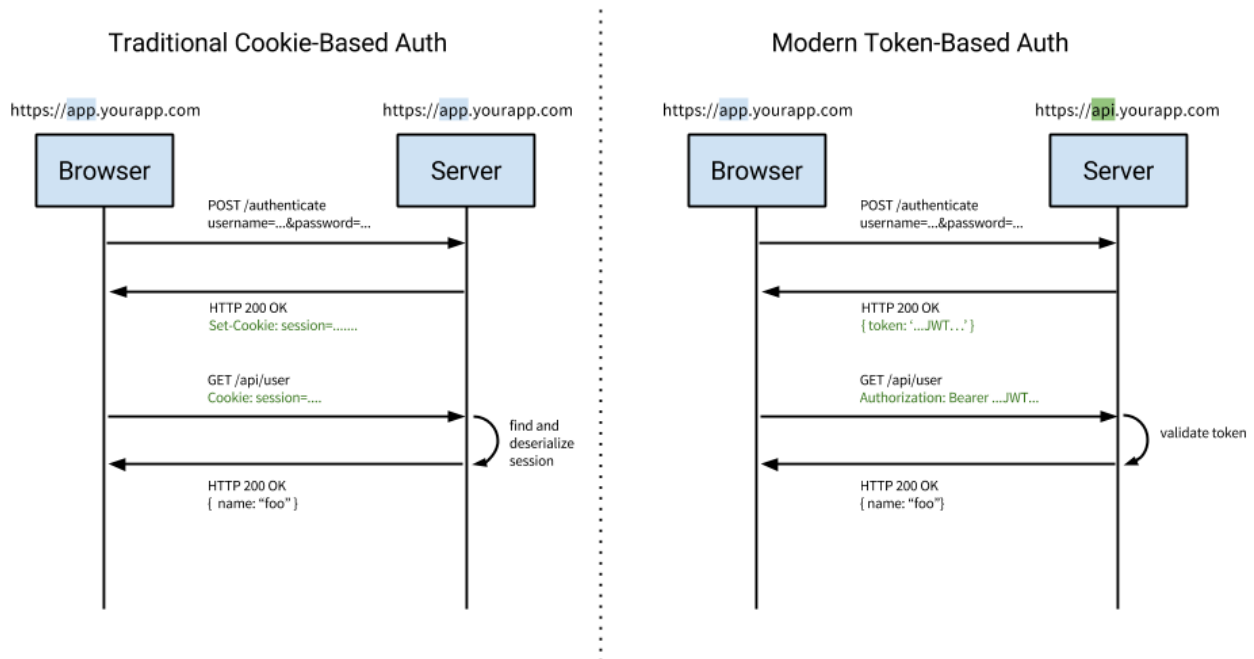


Figure 2 - Cookie and Token based Authentication Mechanisms Comparison

Using token based authentication the applications becomes more scalable, since we don't use session management to store security information we can increase the number of nodes serving the service application to guarantee the maximum availability.

Client-Server Communication

We know that the main parts of the application will run completely apart, and as our view layer will run directly in the clients, that's a security policy that may break the client-server communication.

Modern browser does not allow different hosts direct communication in a web page, so to avoid these problems we need to configure a proxy in our web client.

By installing the module 'http-proxy' with bower commands we can tell our application to redirect the rest calls to a different host that we configure in a JavaScript file.

ATRS-service

The application responsible for handling the requests form ATRS-client and dispatch their results to the database storage and/or handle message processing is the ATRS-service. A fully rest full application that will be developed using spring.

Each component of the rest API will be referenced by a @RestController and mapped by a URL mapping. All the communication will be done through secured https channels and using the JSON format. The use of JSON

eliminates the extra boilerplate process of parsing content to java objects, and these objects then can be persisted in a document-oriented database (Mongo Db).

There are some functional requirements that demand some batch-processing schema, such like:

- send email alerts 48 hours before departure to the clients;
- send email alerts to the staff with the reservation chart of the flight one hour before departure

To accomplish this behavior spring beans with the methods that execute that kind of processing, will be reference in a *ThreadPoolTaskExecutor* to verify the need of any action based on consulting the database.

ATRS-ms

In addition to the functional requirements that we need to accomplish to send mail messages we have a non-functional requirement that requires to each of the alerts have to be guaranteed.

Every alert on the system must be sent to a JMS queue to be handled by another software component. This component is ATRS-ms is responsible to listen to this queue and act always when a new message arrives. If this components breakdowns or becomes instable for some reason, the messages will still been recorded in the JMS queue and will be handled as soon as the application goes back to normal.

Every execution of the methods that handle messages should be done in a transaction context. In case of an execution error an exception will be thrown causing the roll process to be rolled back and the message will then come back to the queue to be processed later.

Detailed View

Programming Languages

The system will be developed fundamentally using the Java programming language in its 8 version. In the view layer we will use HTML, Javascript and CSS.

In the next table, we can see the technologies used in each layer of the application:

Layer	Technology
Data Access	JSON
Business Rules and Services	Java
Presentation	HTML, Javascript and CSS

Communication and Data Storage

The communication between the client application and the server will be realized through secure protocol HTTPS.

The data message exchange between the application running in the browser and the application server will be done with restful services, using JavaScript Object Notation (JSON).

In the server side the communications between the internal layers of the ATRS-service will be done in the primary phase memory, because the objects will be in the same process in the JVM.

There will not be any direct communication between the ATRS-service and the ATRS-ms. These communication will be done through a JMS queue available in a JBoss Application Server.

The data will be stored in a document-based database called Mongo-Db. The components responsible for read/write information in this database will be developed using the library spring-data-mongodb. This library offers the possibility to execute parametrized queries into the database using just annotations in a repository interface.

The access of remote and local files will be implemented using the JAVA NIO API.

Security and Audit

For authorization management the library Spring Security 4.1. The spring beans with restricted operations must be marked with these library annotations to check automatically if the user logged in has sufficient privileges to access the operation.

Secure Sockets Layer (SSL) is the pattern for the communication between client and server.

We will use the Log API SLF4J 1.7.21 (Standard Logging for Java), and the implementation Logback 1.1.7 for the text log..

Design and Coding Patterns

The code should be standardized in confluence to the Oracle *Code Conventions*, available in <http://www.oracle.com/technetwork/java/codeconvtoc-136057.html>.

Logical View

Each component of the ATRS interacts with each other by specific protocols and manners to guarantee the communication efficiency, data security and resilience and scalability of the software. The next figure shows the basic component interaction in the solution:

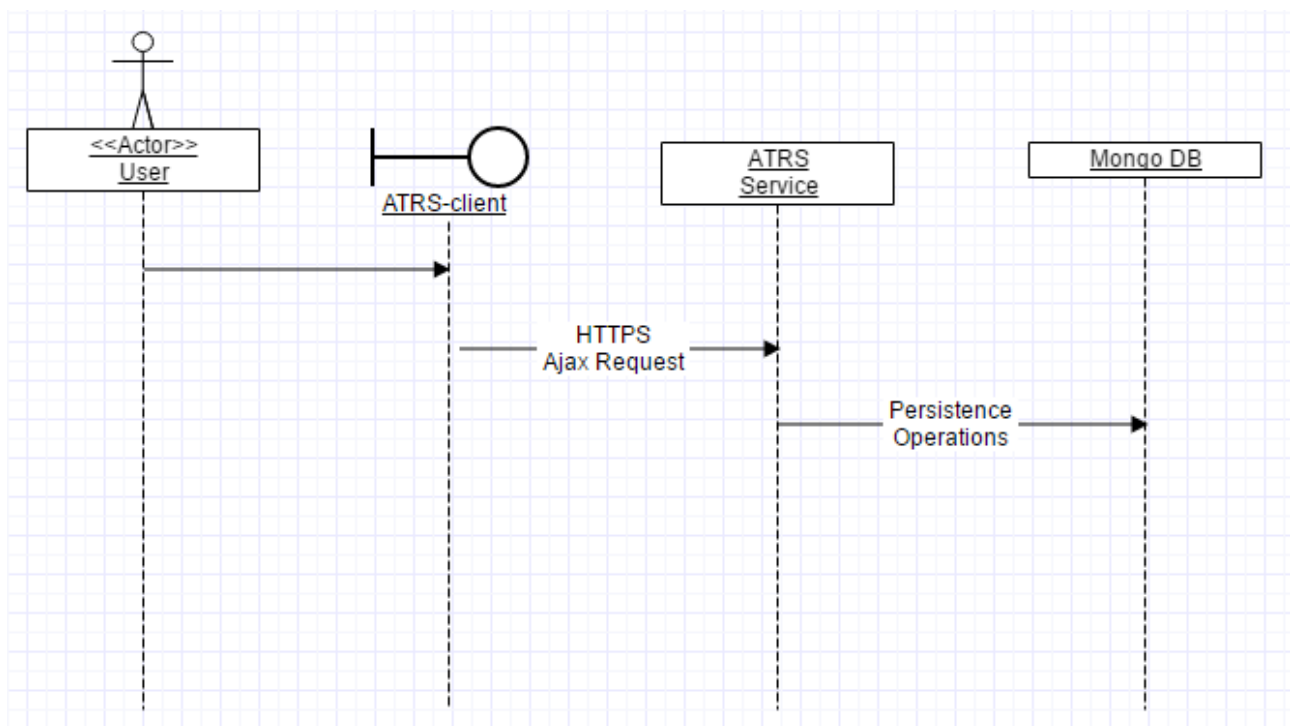


Figure 3 - Basic Component Interaction

In this interaction model, the user will use the interfaces available in ATRS-view to interact with the system. The operations received by the view layer will then be processed by the angular-js controllers and sent to the ATRS-Service rest full web services.

In the rest full web services, spring beans handled by the spring-data-mongodb will so handle all the persistence operations need to process the requisition. When needed beans should add messages to a JMS Queue to be handled by the ATRS-MS.

By separating the components that process are consumed messages guarantee the independence of the process and guarantee the processing of all messages that arrive in the queue. The next figure shows the basic interaction that happens in message processing.

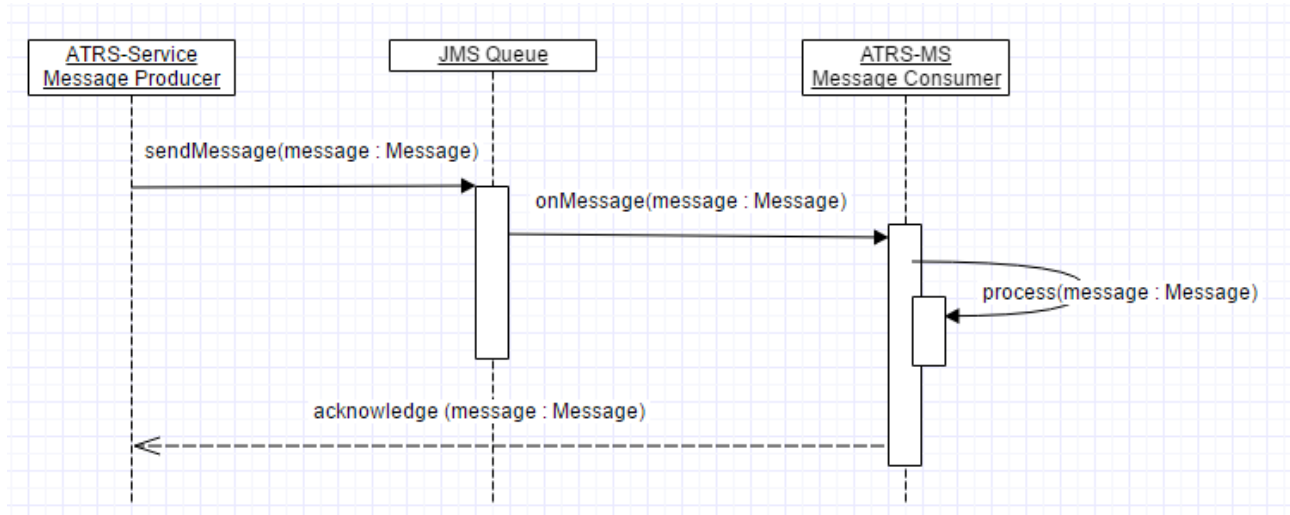


Figure 4 - Message Processing

Data Model

Here we present the basic data model that fulfil the basic requirements of the system:

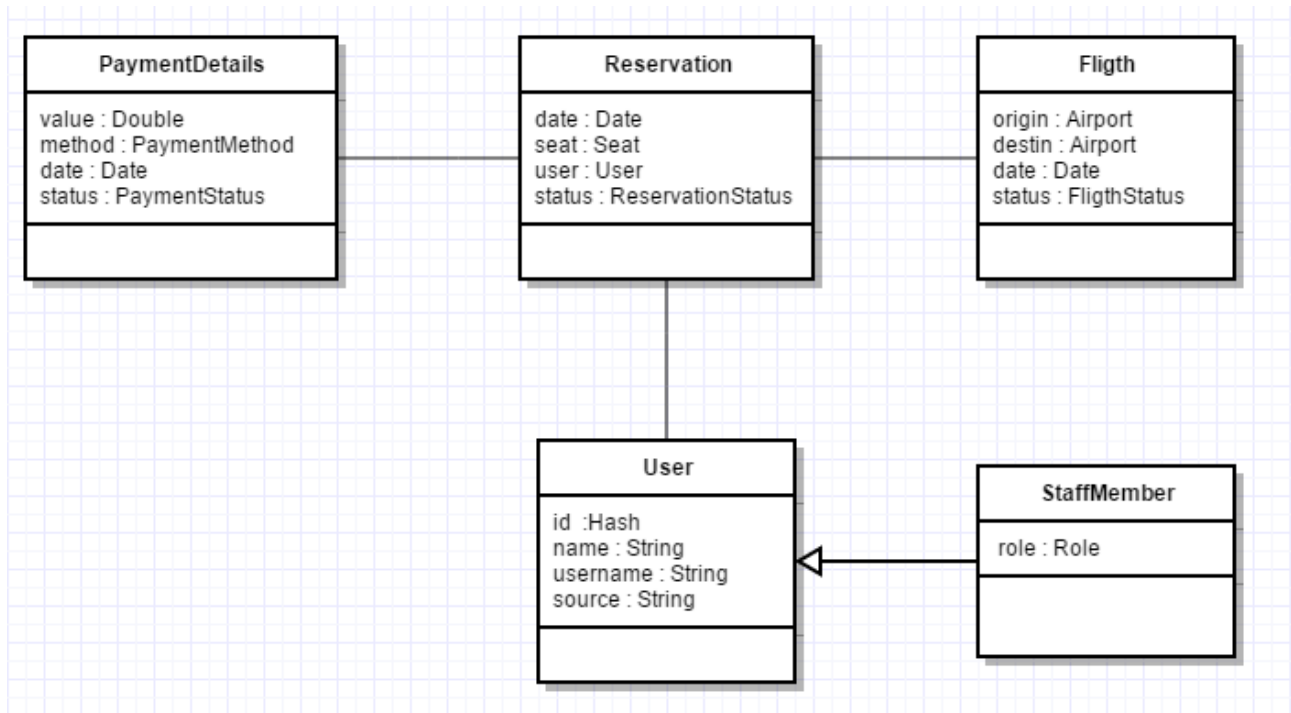
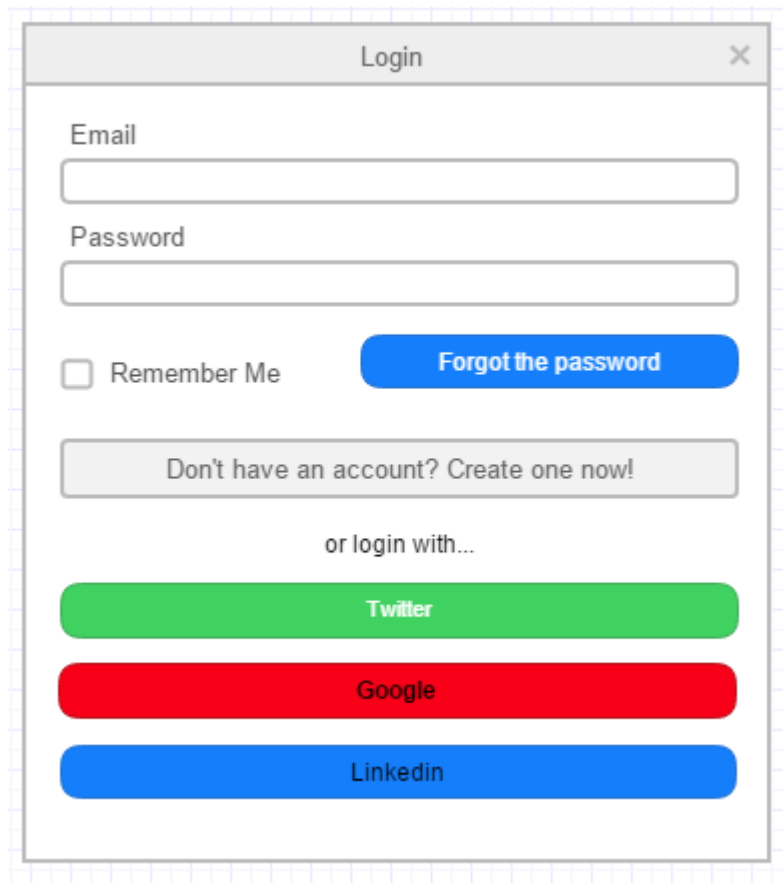


Figure 5 - Basic Data Model

Human Interface Design

In this section we show the main human interface design that accomplish the basic functional requirements of the system.

Login



A login form UI design presented as a wireframe on a light blue grid background. The form is enclosed in a rounded rectangle with a light gray header bar containing the title "Login" and a close button (X). The form contains the following elements: an "Email" label above a text input field; a "Password" label above a text input field; a checkbox labeled "Remember Me"; a blue button labeled "Forgot the password"; a light gray button labeled "Don't have an account? Create one now!"; the text "or login with..."; and three large, rounded, colored buttons for social login: a green button labeled "Twitter", a red button labeled "Google", and a blue button labeled "LinkedIn".

Sign up

Sign Up

First Name

Last Name

Email Address

Password

Confirm Password

☒ I agree to the [Terms of Service](#)

☒ Keep me updated on new features

SIGN UP

