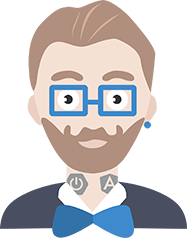
****

*Creating Microservices with JHipster*

(Starter guide)

*Ravi Kalla*

** [**](https://www.certmetrics.com/amazon/public/badge.aspx?i=1&t=c&d=2018-01-20&ci=AWS00406237&dm=80)

*Email:* [*ravi2523096@gmail.com*](mailto:ravi2523096@gmail.com)

*Linkedin:* [*https://www.linkedin.com/in/ravikalla/*](https://www.linkedin.com/in/ravikalla/)

*Github:* [*https://github.com/ravikalla*](https://github.com/ravikalla)

*M# +1-585-281-9787*

**Creating microservices using Jhipster**

**What is JHipster?**

[JHipster](https://www.jhipster.tech/) is a code generator tool that incorporates best design patterns and supports cutting-edge development tools and platforms. The generated code looks like a handwritten code for developers and is not dependent on any proprietary tools (I have seen couple of code generation tools in the past which generates complex code and it was very difficult to understand and maintain).

In my opinion, it would take couple of months for any senior developer to create a handwritten project from scratch with all such good design patterns in it.

**Main components and tools:**

* [SpringBoot](https://projects.spring.io/spring-boot/) – all applications are created in SpringBoot format
* [Angular6](https://angular.io/) and [React](https://reactjs.org/) – for User Interface
* [Yeoman](http://yeoman.io/) – front-end scaffolding tool
* [Swagger](http://swagger.io/) – API Documentation
* [Maven](https://maven.apache.org/), [Gradle](https://gradle.org/), [Npm](https://yarnpkg.com/en/), [Gulp](http://gulpjs.com/), [Bower](https://bower.io/) - dependency managers and build tools
* [Cucumber](https://cucumber.io/), [Gatling](http://gatling.io/), [Protractor](https://www.protractortest.org/#/), [Jasmine](https://jasmine.github.io/) – test frameworks
* [Liquibase](https://www.liquibase.org/) – DB Versioning tools
* SQL (MySQL, MariaDB, PostgreSQL, Oracle, SQL Server), MongoDB, Couchbase, Cassandra – Databases
* Jenkins, TravisCI, CircleCI, Gitlab – Continuous Integration tools

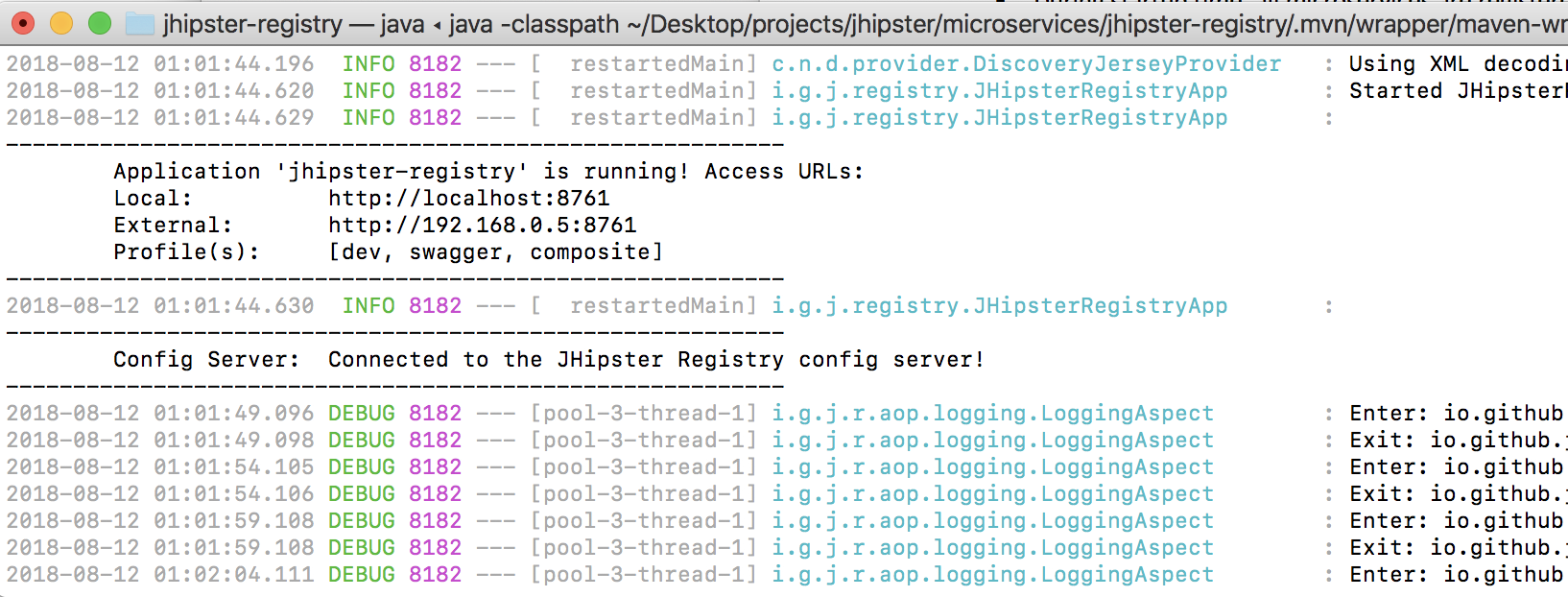
**Prerequisites:**

* Install Java (JDK 8) – [Link](http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html)
* Install Maven – [Link](https://maven.apache.org/install.html)
* Install Jhipster– [Link](https://www.jhipster.tech/installation/)

**Create Microservices application:**

1. **Service Registry:**
   * + During startup time, all microservices are registered in this registry application with a specific name.
     + Code in “Microservice A” can trigger webservices of “Microservice B” just by mentioning the name of “Microservice B” instead of mentioning the IP address of the “Microservice B”.
     + Common examples are “Eureka”, “Consul”, “Zookeeper”.
     + As I am using JHipster in this case, I am using Jhipster Registry which is a flavor of Eureka config server.

|  |
| --- |
| **Console Commands for Service Registry setup** |
| * mkdir ~/microservices/ * cd ~/microservices/ * git clone <https://github.com/jhipster/jhipster-registry> * cd jhipster-registry * ./mvnw |

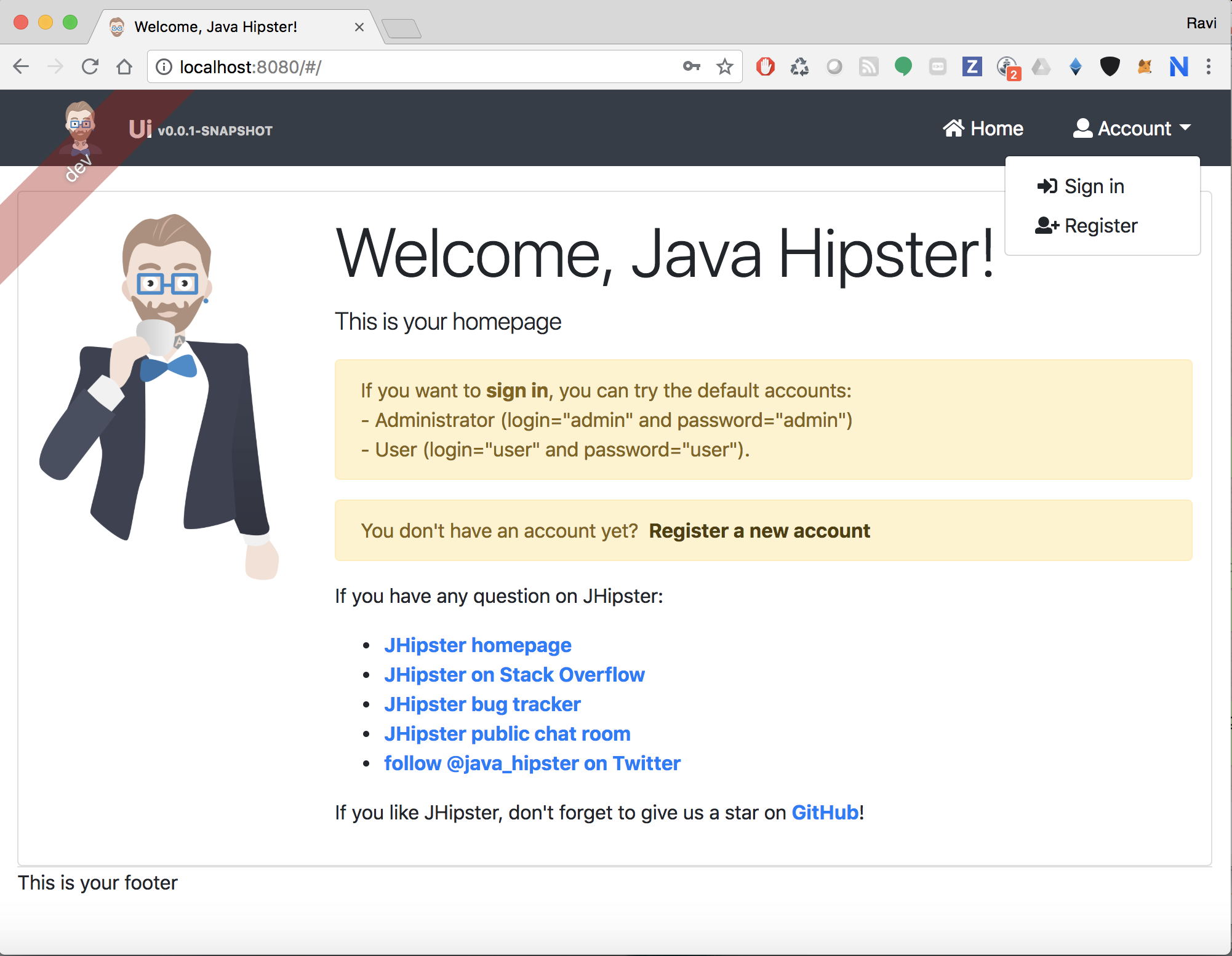


**URL:** <http://localhost:8761/>

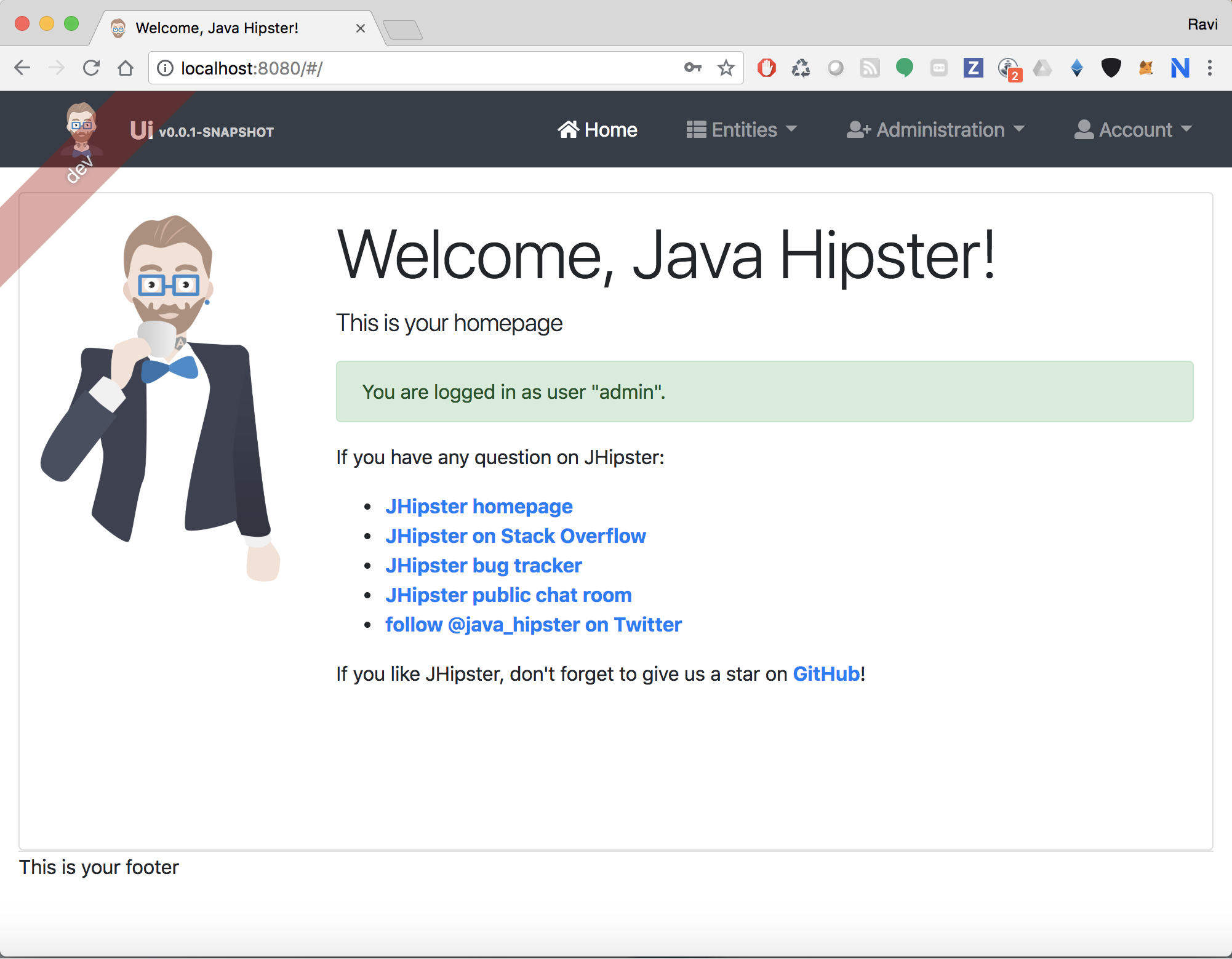
1. **Create UI (customer-service):**
   * + Create another project using ‘yo jhipster’ and select the project type as ‘Microservice gateway’. This is where all UI code lies.

|  |
| --- |
| **Console Commands for “customer-service” microservice creation** |
| * mkdir ~/microservices/ui * cd ~/microservices/ui |
| * yo jhipster   ? Which \*type\* of application would you like to create? Microservice gateway  ? What is the base name of your application? ui  ? As you are running in a microservice architecture, on which port would like your server to run? It should be uniqu  e to avoid port conflicts. 8080  ? What is your default Java package name? in.ravikalla.microservices.ui  ? Which service discovery server do you want to use? JHipster Registry (uses Eureka, provides Spring Cloud Config su  pport and monitoring dashboards)  ? Which \*type\* of authentication would you like to use? JWT authentication (stateless, with a token)  ? Which \*type\* of database would you like to use? SQL (H2, MySQL, MariaDB, PostgreSQL, Oracle, MSSQL)  ? Which \*production\* database would you like to use? Microsoft SQL Server  ? Which \*development\* database would you like to use? H2 with in-memory persistence  ? Do you want to use Hibernate 2nd level cache? Yes  ? Would you like to use Maven or Gradle for building the backend? Maven  ? Which other technologies would you like to use?  ? Which \*Framework\* would you like to use for the client? Angular 6  ? Would you like to enable \*SASS\* support using the LibSass stylesheet preprocessor? No  ? Would you like to enable internationalization support? No  ? Besides JUnit and Jest, which testing frameworks would you like to use? Cucumber  ? Would you like to install other generators from the JHipster Marketplace? No |

**URL:** <http://localhost:8080/>



**Login ID / Password:** admin / admin



1. **Microservice1 (savings account):**
   * + Create a simple microservice with command ‘yo jhipster’. This microservice is a simple Spring Boot application and can register itself in Service Registry.
     + Create two entities called as ‘savingsaccount’ and ‘transaction’ using Jhipster
     + Create One-To-Many relationship between ‘savingsaccount’ and ‘transaction’ entities.

|  |
| --- |
| **Console Commands for “savings-acc” microservice creation** |
| * mkdir ~/microservices/savings-acc * cd ~/microservices/savings-acc |
| * yo jhipster   ? Which \*type\* of application would you like to create? Microservice application  ? What is the base name of your application? savingsaccount  ? As you are running in a microservice architecture, on which port would like your server to run? It should be uniqu  e to avoid port conflicts. 8081  ? What is your default Java package name? in.ravikalla.microservices.savingsaccount  ? Which service discovery server do you want to use? JHipster Registry (uses Eureka, provides Spring Cloud Config support and monitoring dashboards)  ? Which \*type\* of authentication would you like to use? JWT authentication (stateless, with a token)  ? Which \*type\* of database would you like to use? SQL (H2, MySQL, MariaDB, PostgreSQL, Oracle, MSSQL)  ? Which \*production\* database would you like to use? Microsoft SQL Server  ? Which \*development\* database would you like to use? H2 with in-memory persistence  ? Do you want to use the Spring cache abstraction? Yes, with the Hazelcast implementation (distributed cache, for mu  ltiple nodes)  ? Do you want to use Hibernate 2nd level cache? Yes  ? Would you like to use Maven or Gradle for building the backend? Maven  ? Which other technologies would you like to use?  ? Would you like to enable internationalization support? No  ? Besides JUnit and Jest, which testing frameworks would you like to use? Cucumber  ? Would you like to install other generators from the JHipster Marketplace? No |
| * yo jhipster:entity savingsaccount   Generating field #1  ? Do you want to add a field to your entity? Yes  ? What is the name of your field? accountnumber  ? What is the type of your field? Integer  ? Do you want to add validation rules to your field? Yes  ? Which validation rules do you want to add? Required  Generating field #2  ? Do you want to add a field to your entity? Yes  ? What is the name of your field? accountname  ? What is the type of your field? String  ? Do you want to add validation rules to your field? Yes  ? Which validation rules do you want to add? Required, Minimum length, Maximum length  ? What is the minimum length of your field? 1  ? What is the maximum length of your field? 100  Generating field #3  ? Do you want to add a field to your entity? Yes  ? What is the name of your field? amount  ? What is the type of your field? Double  ? Do you want to add validation rules to your field? Yes  ? Which validation rules do you want to add? Required  Generating field #4  ? Do you want to add a field to your entity? Yes  ? What is the name of your field? generalinfo  ? What is the type of your field? String  ? Do you want to add validation rules to your field? No  Generating field #5  ? Do you want to add a field to your entity? No  Generating relationships to other entities  ? Do you want to add a relationship to another entity? No  ? Do you want to use separate service class for your business logic? Yes, generate a separate service interface and implementation  ? Do you want to use a Data Transfer Object (DTO)? [BETA] Yes, generate a DTO with MapStruct  ? Do you want to add filtering? Dynamic filtering for the entities with JPA Static metamodel  ? Do you want pagination on your entity? Yes, with pagination links |
| * yo jhipster:entity transaction   Generating field #1  ? Do you want to add a field to your entity? Yes  ? What is the name of your field? amount  ? What is the type of your field? Double  ? Do you want to add validation rules to your field? Yes  ? Which validation rules do you want to add? Required, Minimum, Maximum  ? What is the minimum of your field? -100000000000  ? What is the maximum of your field? 100000000000  Generating field #2  ? Do you want to add a field to your entity? Yes  ? What is the name of your field? transactiondate  ? What is the type of your field? LocalDate  ? Do you want to add validation rules to your field? Yes  ? Which validation rules do you want to add? Required  Generating field #3  ? Do you want to add a field to your entity? No  Generating relationships to other entities  ? Do you want to add a relationship to another entity? Yes  ? What is the name of the other entity? savingsaccount  ? What is the name of the relationship? savingsaccount  ? What is the type of the relationship? many-to-one  ? When you display this relationship on client-side, which field from 'savingsaccount' do you want to use? This field will be displayed as a String, so it cannot be a Blob accountnumber  ? Do you want to add any validation rules to this relationship? No  Generating relationships to other entities  ? Do you want to add a relationship to another entity? No  ? Do you want to use separate service class for your business logic? Yes, generate a separate service interface and implementation  ? Do you want to use a Data Transfer Object (DTO)? [BETA] Yes, generate a DTO with MapStruct  ? Do you want to add filtering? Dynamic filtering for the entities with JPA Static metamodel  ? Do you want pagination on your entity? Yes, with pagination links |

**Endpoint URL:** <https://localhost:8081>

**Note:** As this microservice is just for hosting some webservices and business logic, there is no useful UI even if you hit above URL.

1. **Microservice2 (customer service):**
   * + Create another microservice as it was done in previous step
     + We created entity called as ‘appointment’ using Jhipster

|  |
| --- |
| **Console Commands for “customer-service” microservice creation** |
| * mkdir ~/microservices/customer-service * cd ~/microservices/customer-service |
| * yo jhipster   ? Which \*type\* of application would you like to create? Microservice application  ? What is the base name of your application? customerservice  ? As you are running in a microservice architecture, on which port would like your server to run? It should be uniqu  e to avoid port conflicts. 8082  ? What is your default Java package name? in.ravikalla.microservices.customerservice  ? Which service discovery server do you want to use? JHipster Registry (uses Eureka, provides Spring Cloud Config su  pport and monitoring dashboards)  ? Which \*type\* of authentication would you like to use? JWT authentication (stateless, with a token)  ? Which \*type\* of database would you like to use? SQL (H2, MySQL, MariaDB, PostgreSQL, Oracle, MSSQL)  ? Which \*production\* database would you like to use? Microsoft SQL Server  ? Which \*development\* database would you like to use? H2 with in-memory persistence  ? Do you want to use the Spring cache abstraction? Yes, with the Hazelcast implementation (distributed cache, for mu  ltiple nodes)  ? Do you want to use Hibernate 2nd level cache? Yes  ? Would you like to use Maven or Gradle for building the backend? Maven  ? Which other technologies would you like to use?  ? Would you like to enable internationalization support? No  ? Besides JUnit and Jest, which testing frameworks would you like to use? Cucumber  ? Would you like to install other generators from the JHipster Marketplace? No |
| * yo jhipster:entity appointment   Generating field #1  ? Do you want to add a field to your entity? Yes  ? What is the name of your field? visitorname  ? What is the type of your field? String  ? Do you want to add validation rules to your field? Yes  ? Which validation rules do you want to add? Required  Generating field #2  ? Do you want to add a field to your entity? Yes  ? What is the name of your field? time  ? What is the type of your field? ZonedDateTime  ? Do you want to add validation rules to your field? Yes  ? Which validation rules do you want to add? Required  Generating field #3  ? Do you want to add a field to your entity? Yes  ? What is the name of your field? comments  ? What is the type of your field? String  ? Do you want to add validation rules to your field? No  Generating field #4  ? Do you want to add a field to your entity? No  Generating relationships to other entities  ? Do you want to add a relationship to another entity? No  ? Do you want to use separate service class for your business logic? Yes, generate a separate service interface and  implementation  ? Do you want to use a Data Transfer Object (DTO)? [BETA] Yes, generate a DTO with MapStruct  ? Do you want to add filtering? Dynamic filtering for the entities with JPA Static metamodel  ? Do you want pagination on your entity? Yes, with pagination links |

**Endpoint URL:** <https://localhost:8082>

**Note:** As this microservice is just for hosting some webservices and business logic, there is no useful UI even if you hit above URL.

1. **Create entities in UI (customer-service):**
   * + Try to create entities with names ‘savingsaccount’ ‘transaction’ and ‘appointment’ in the two microservices that were previously created.
     + Jhipster will ask if you want to import entities from other projects. Then, mention the relative paths of the ‘customer service’ and ‘savings account’ microservices that were created in earlier steps.

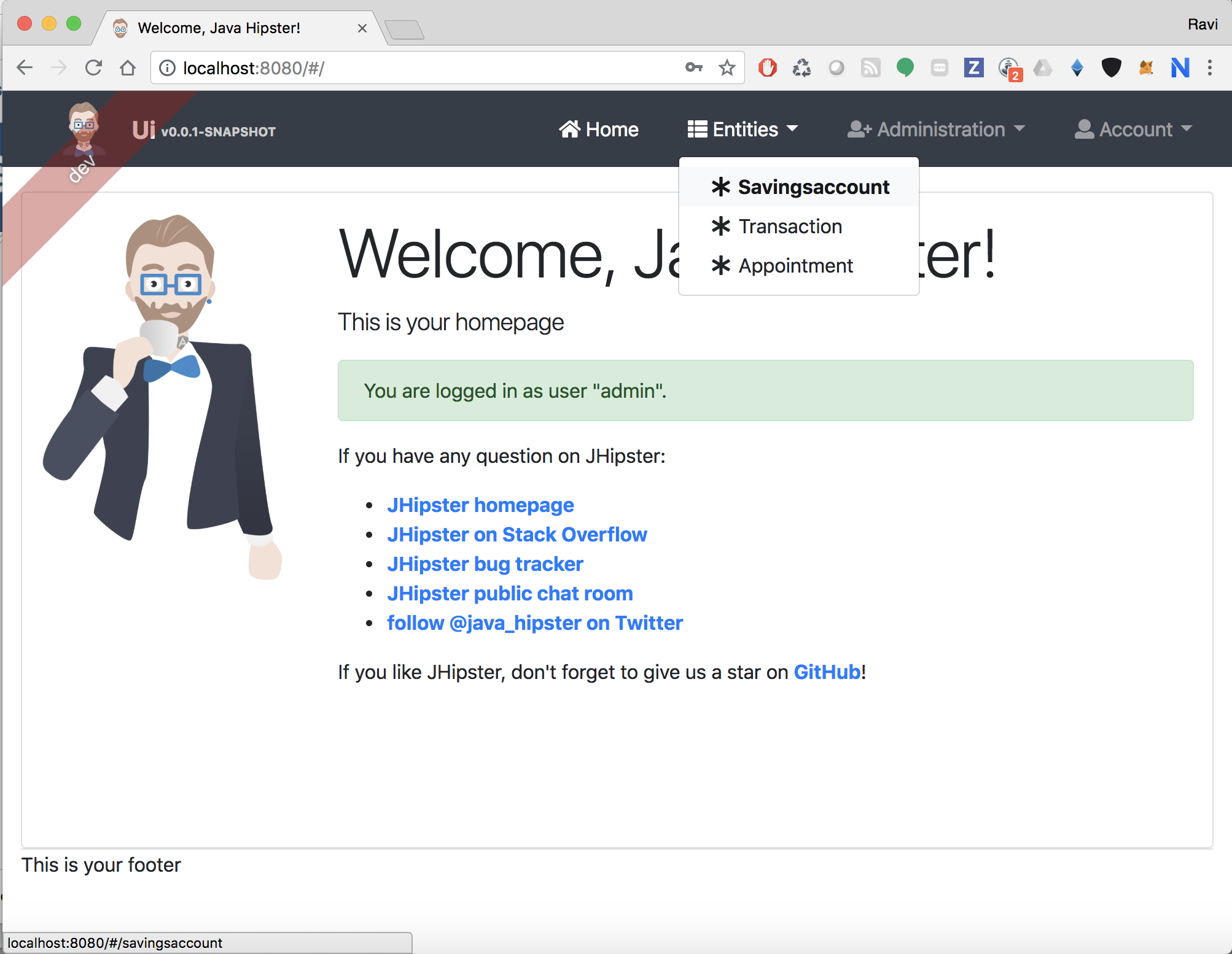
|  |
| --- |
| **Console Commands for importing entities in UI** |
| * yo jhipster:entity savingsaccount   ? Do you want to generate this entity from an existing microservice? Yes  ? Enter the path to the microservice root directory: ../savings-acc |
| * yo jhipster:entity transaction   ? Do you want to generate this entity from an existing microservice? Yes  ? Enter the path to the microservice root directory: ../savings-acc |
| * yo jhipster:entity appointment   ? Do you want to generate this entity from an existing microservice? Yes  ? Enter the path to the microservice root directory: ../customer-service |

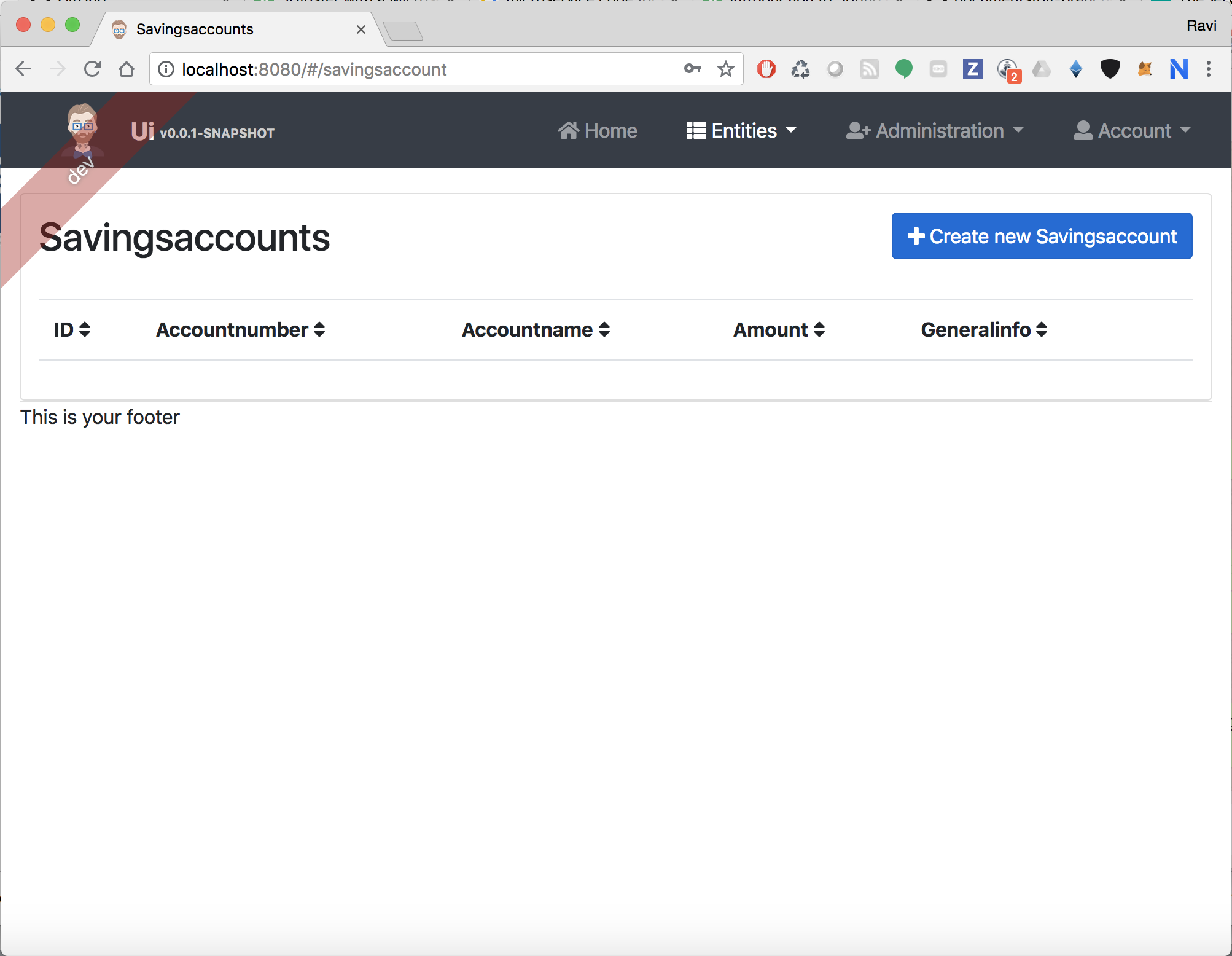
1. **Start all services –**

|  |
| --- |
| **Console Commands for starting all microservices (execute in 4 different terminal windows)** |
| **Start Service Registry:**   * ~/microservices/ jhipster-registry/mvnw   **Start Microservice1(Savings Account):**   * ~/microservices/ savings-acc /mvnw   **Start Microservice2(Start Customer Service):**   * ~/microservices/customer-service/mvnw   **Start UI:**   * ~/microservices/ui/mvnw |

**Application URL:** <https://localhost:8080>

Verify if the imported entities are listed under ‘Entities’ tab as below –

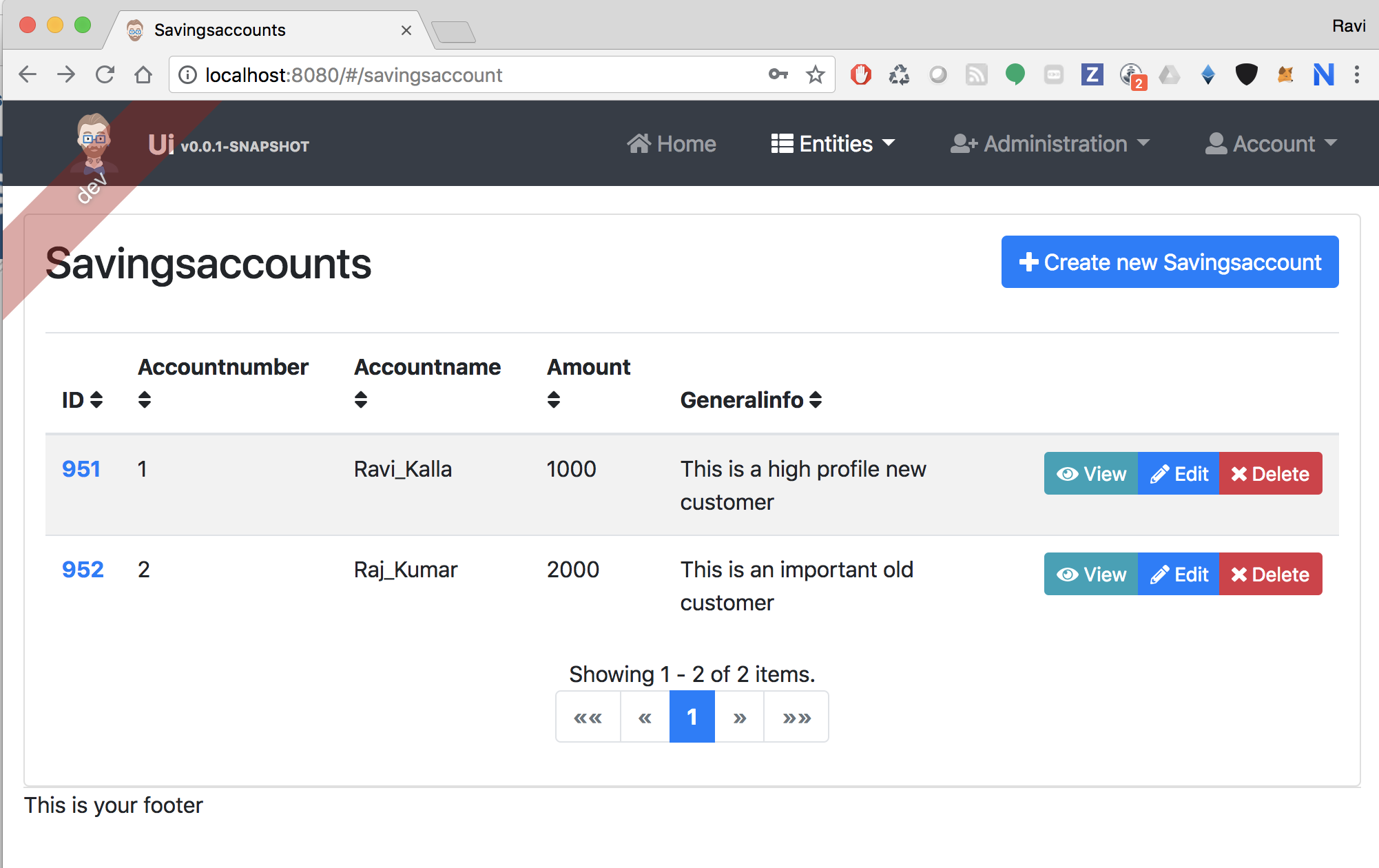


Open “Savingsaccounts” entity screen and notice that there are no entities –  


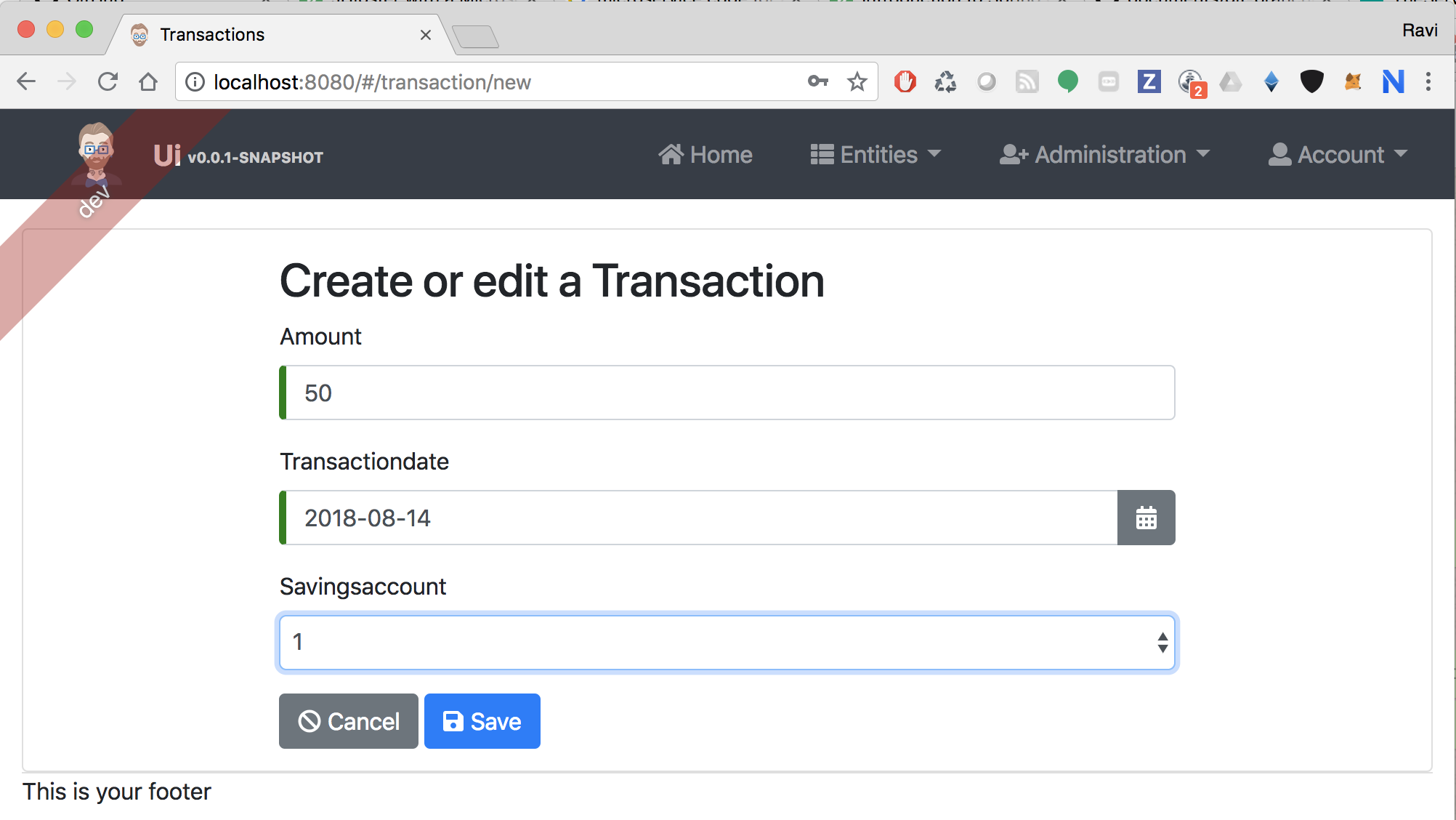
Try to add a new entity by clicking “Create new Savingsaccount” button:



Add couple of entities:



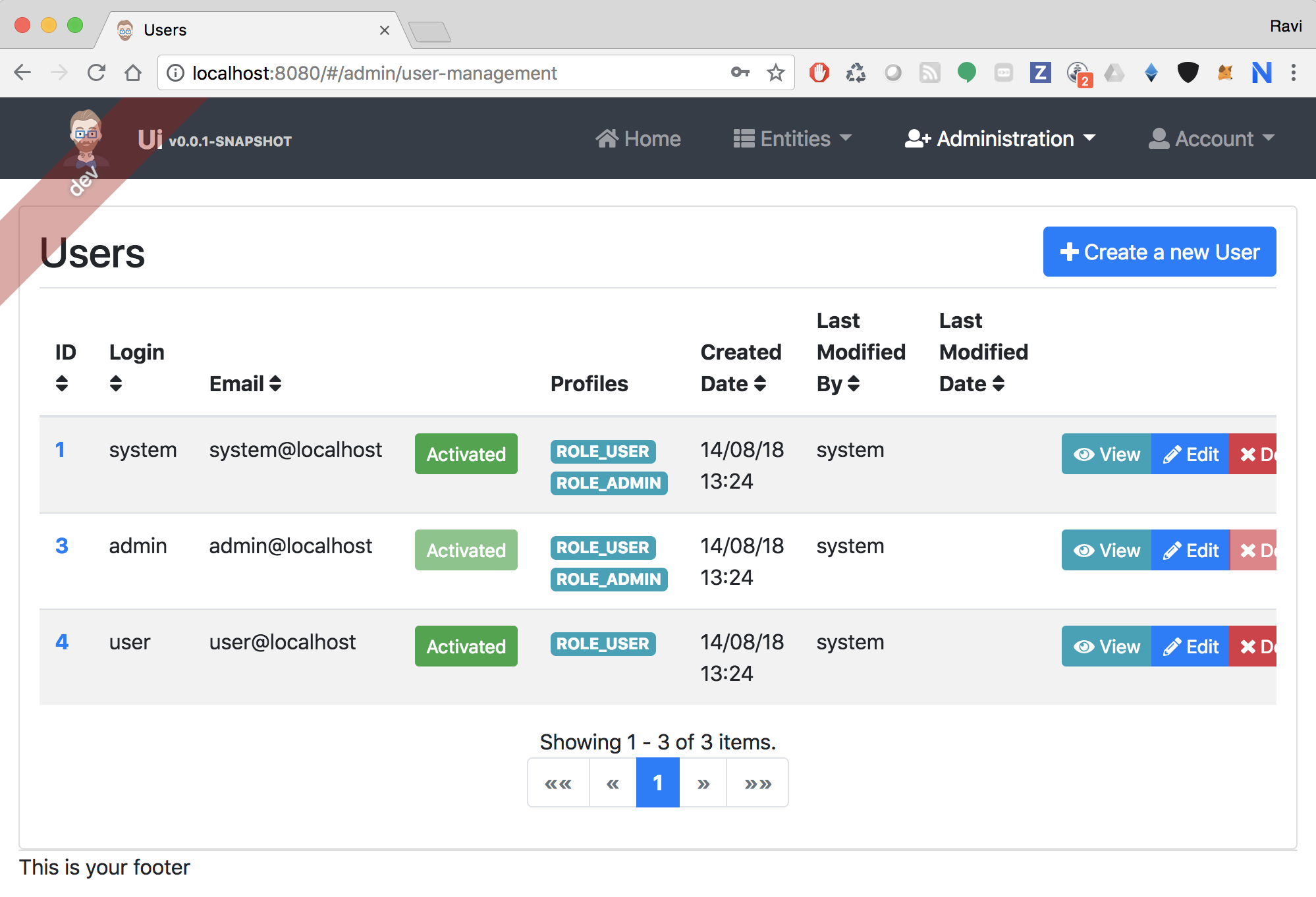
Notice that while creating Transaction entity, you can see dependent entity (Savingsaccount) in a list box at the bottom:



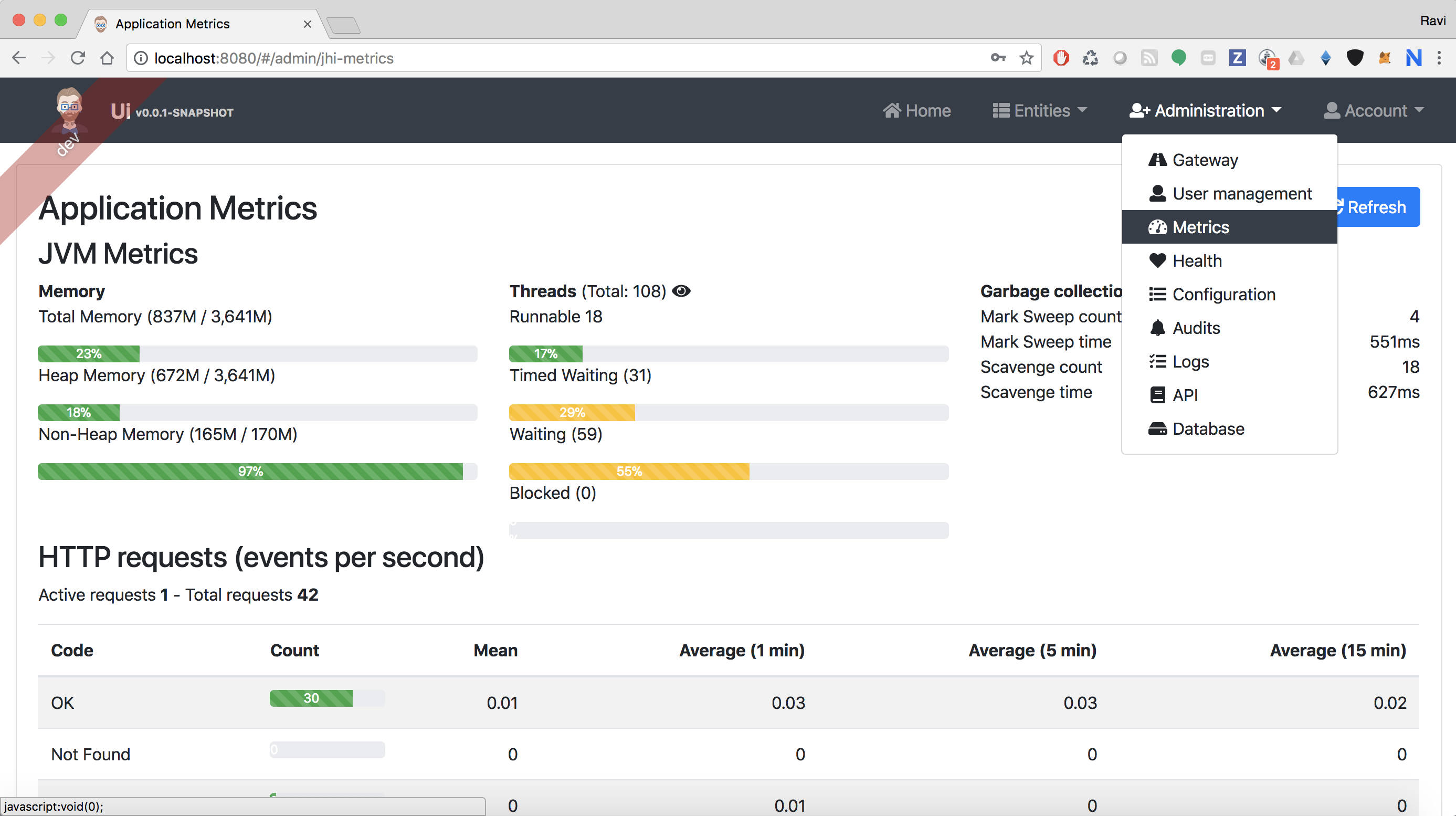
‘Gateway’ page in ‘Administration’ tab contains list of microservices that are registered in the Service Discovery (Eureka server):



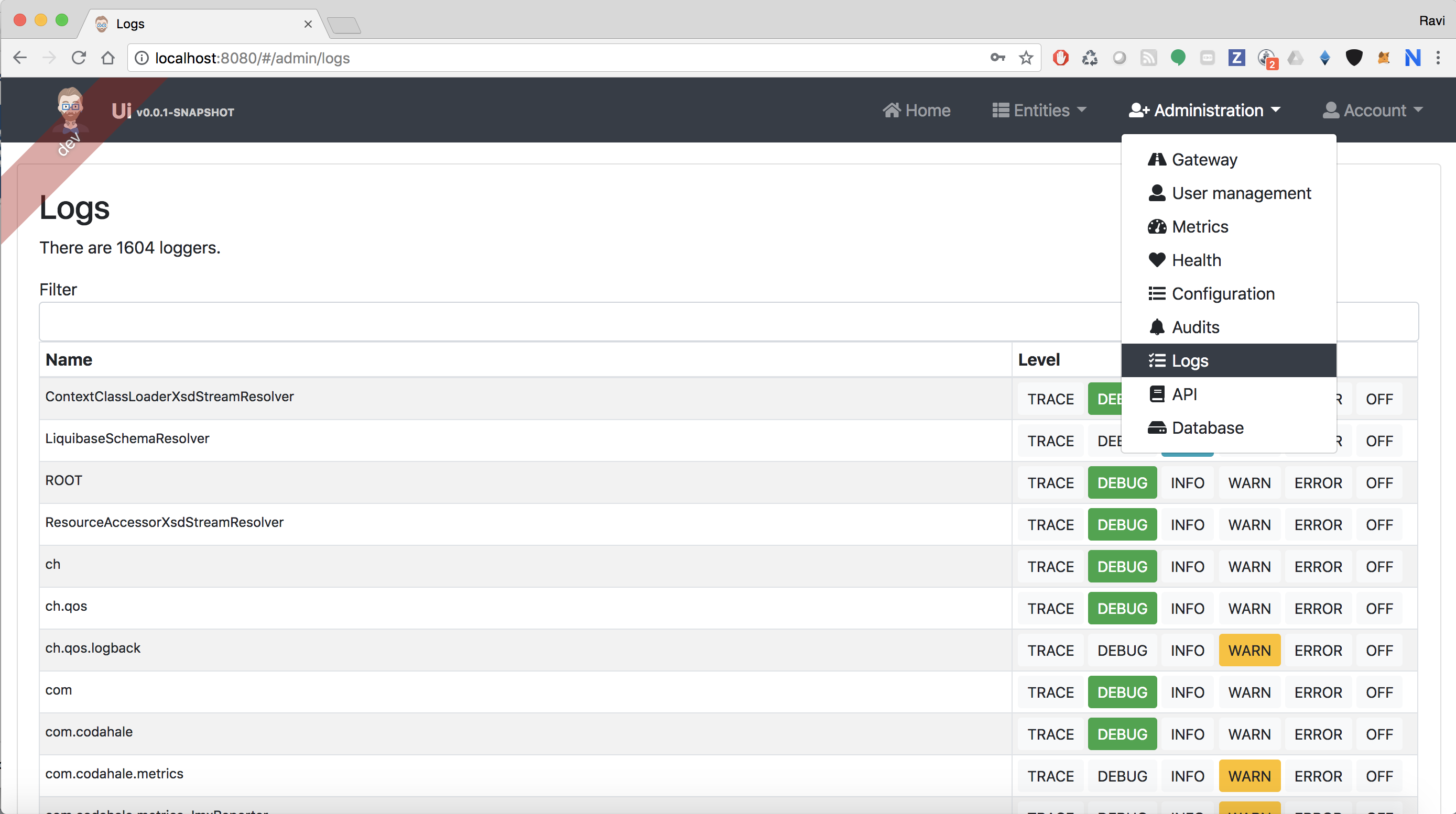
‘Users’ page under ‘Administration’ tab contains the list of registered users in the application:



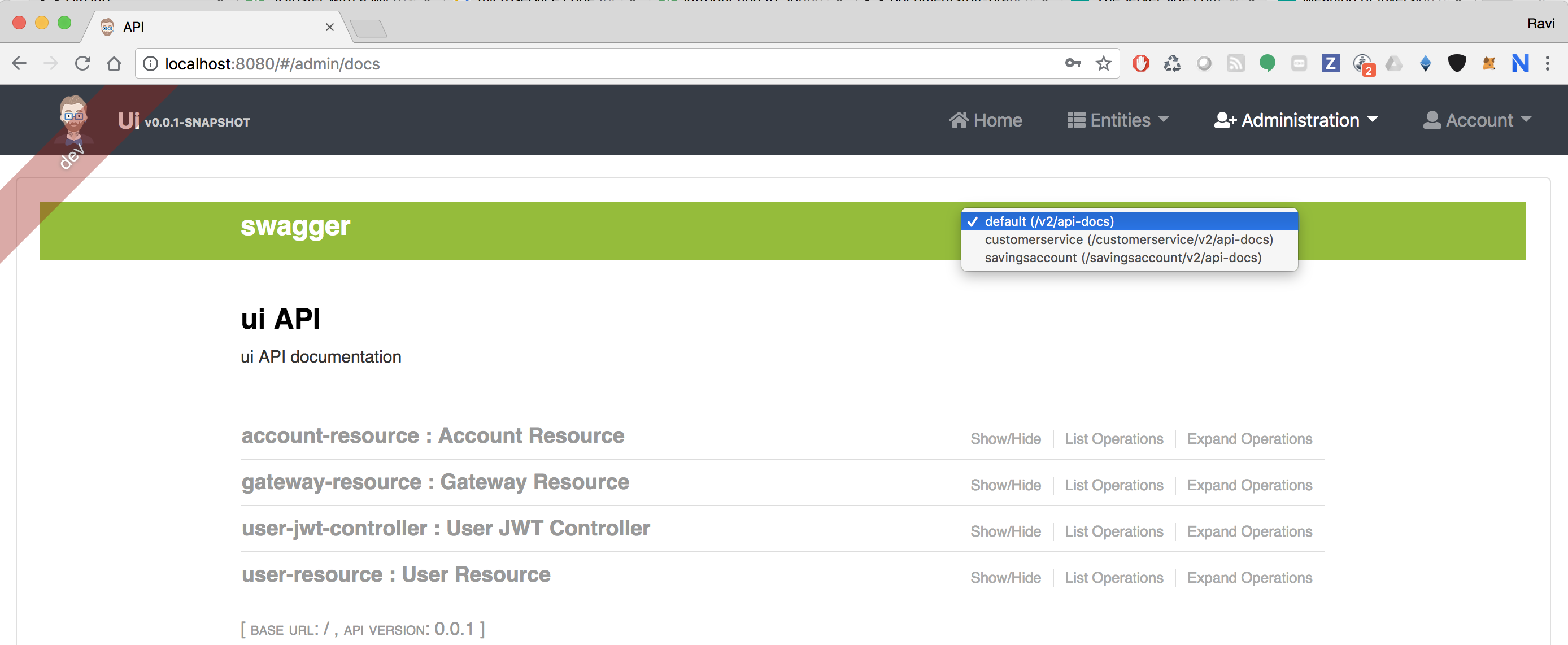
‘Metrics’ page under ‘Administration’ tab contains Application metrics that are obtained using JMX:



‘Logs’ page under ‘Administration’ tab provides dynamic log level configuration capabilities (with Log4J support):



‘Swagger’ page under ‘Administration’ tab provides API Documentation for all microservices:



Autogenerated project in Github: <https://github.com/ravikalla/microservices>