Architectural Pattern for RESTful Service Coordination

16th Workshop "Software Engineering Education and Reverse Engineering"

Agon Memeti, Betim Çiço

August 26th 2016

Outline

- Motivation & Problem Statement
- Contribution
- Reuse Approach (REST Architecture)
- Proposed Approach
- Coordination Model / Framework
- Framework Implementation
- Conclusions & Future Work

Motivation

- Service integration
  - A long-standing research problem
  - Uses software or computer system architectural principles
  - Biggest challenge: linking in-house services (applications) of a single organization in order to share data
- Authorization Problems
  - Should be handled separately from the implementation
- **The goal**: define a suitable architecture for integrating in-house services

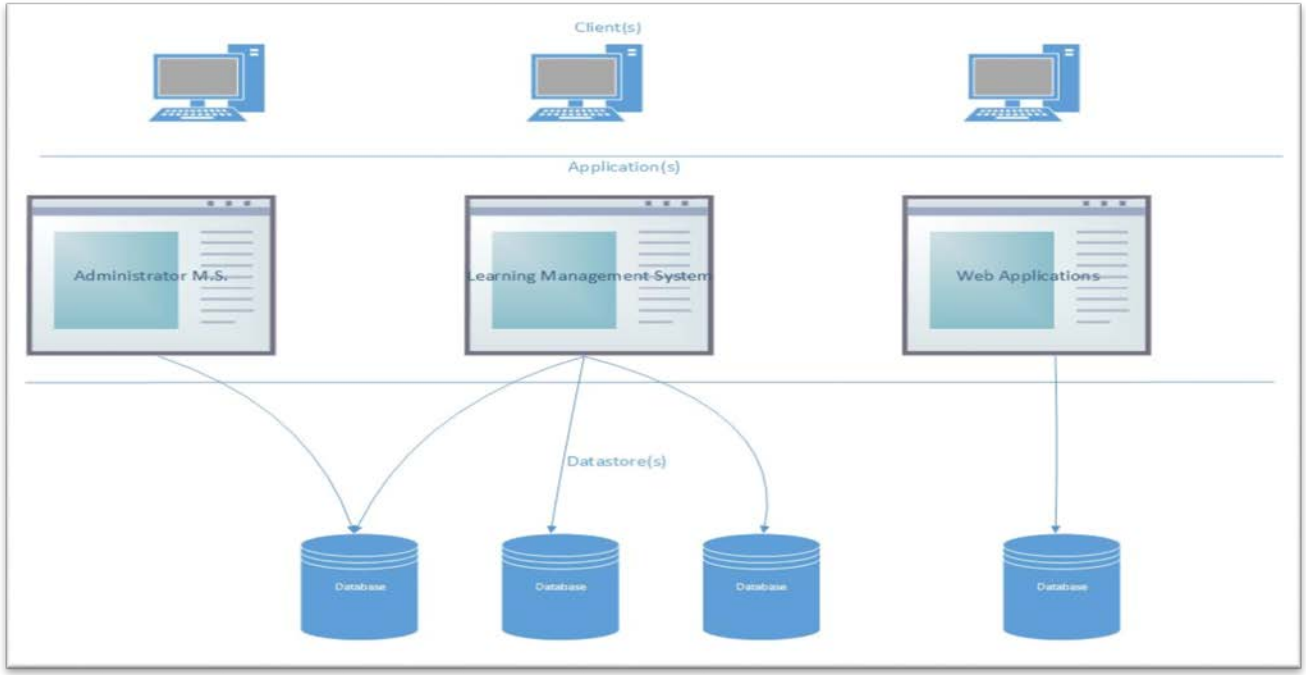
Motivation

- How to integrate the existing in-house applications, reusing existing in-house services for reduced service dependency and increased service flexibility?

- Taking into account the flexibility, reliability and high availability of e-services.

Problem Statement

- Different web application platforms that serve to offer services to users but not integrated
  - Difficult to evolve an existing service
  - Permissions required for using the service and managed by service creator



Contribution

- **Proposed** a model / framework for integration of in-house services
  - **decoupling authorization** concerns from the implementation
  - **reduced service dependency**
  - **flexible integration** of registered in-house services such as University Services.

- **Validated** the model by implementing the framework and applying on a case study

Approach

1. Integrate in-house applications in order to share data
   - Cloud-based, use existing SOA approaches
2. Export data as services
   - The granularity of the exported data is important for reuse
   - Services should match the business concepts
   - RESTful services – work with resources instead of operations.
3. Propose a model
   - Integrated authentication (based on OAuth 2.0 principles)
   - Decoupled authorization from implementation
   - Improved interoperability and flexibility
4. Validate the model
   - Implement a framework
   - Case study

Reuse Approach

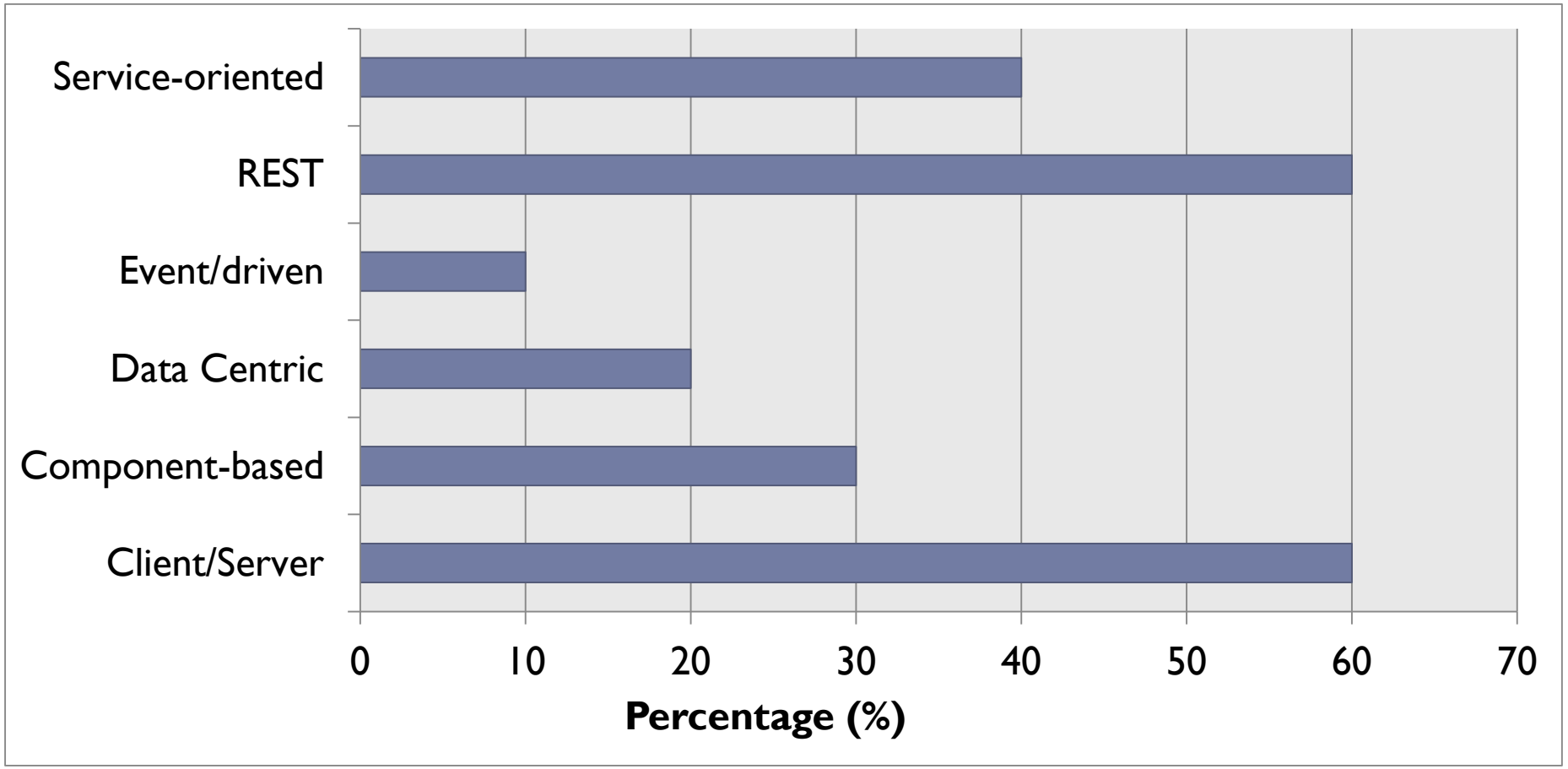
- **Reusability -** is a primary attraction for developers when discussing about reusing existing services.

- Enable providers and developers of a system to port their services, enabling user communities to evolve.

- Three levels of reuse
  - Federation
  - Domain
  - **Application**

Reuse Approach (2)

- Percentage Architectural Styles used by Software Companies

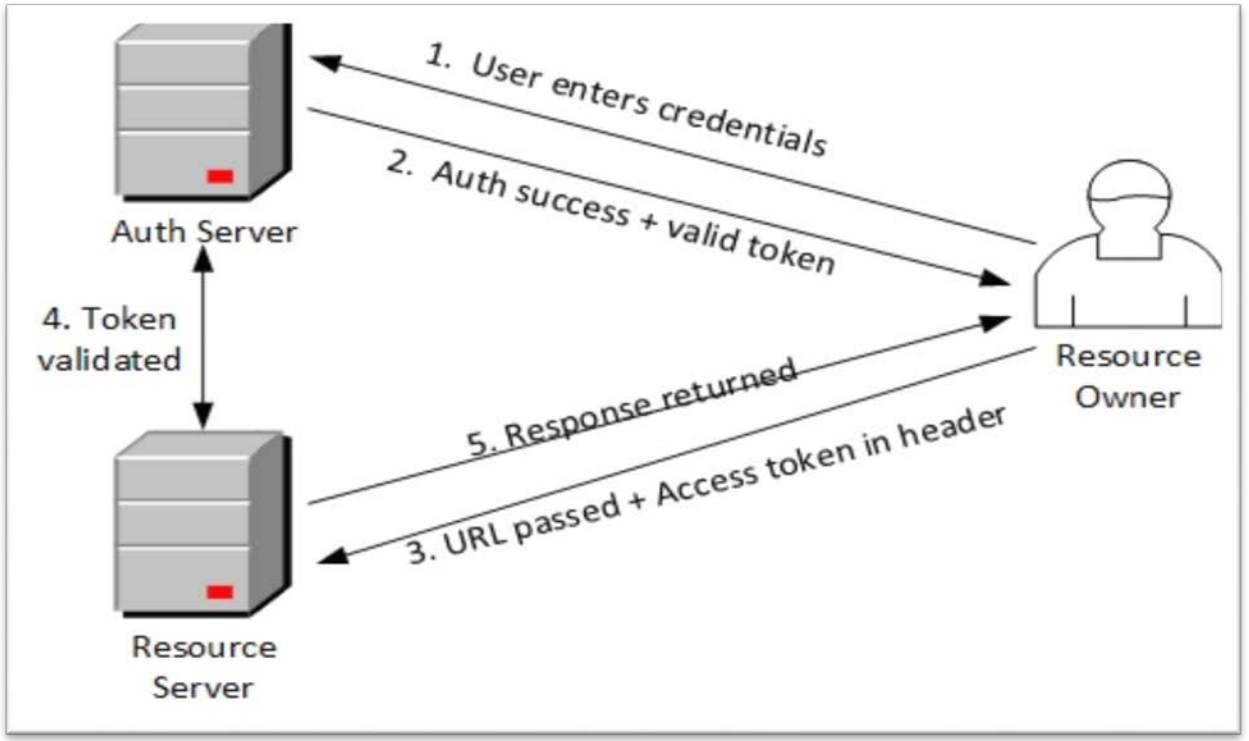


Proposed Approach (REST & OAuth 2.0)

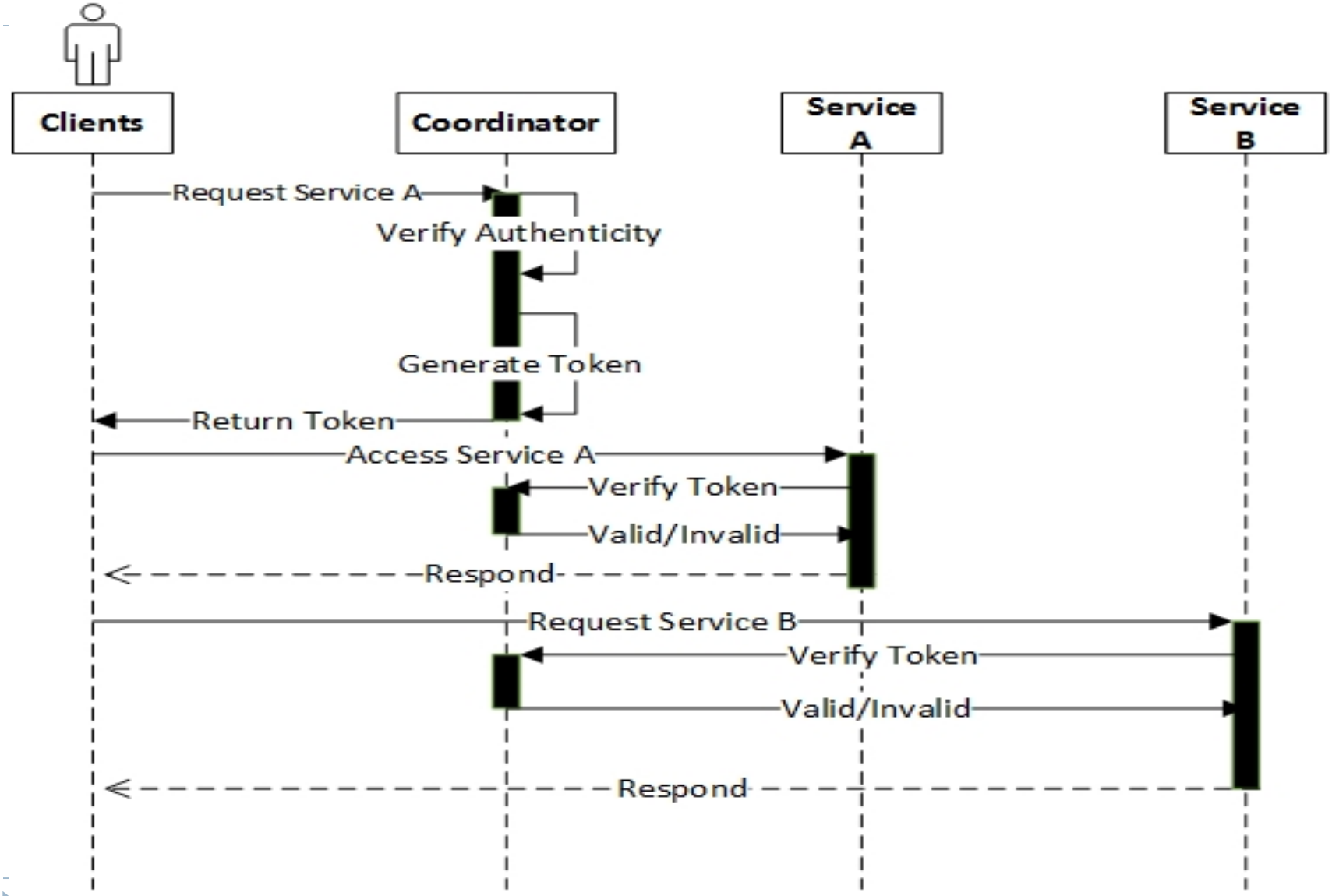
- **RESTful Services [Roy Fielding – PhD Thesis]**
- Architectural style (collection of principles), lighter than SOAP-based Web Services, due to their simplicity, heterogeneity and web-based format.
- Simplifying usage, development, and deployment to the web.
- HTTP verbs are used for different operations:
  - GET returns the list of resources.
  - POST creates a new resource. Data is provided in the body.
  - PUT updates an existing resource.
  - PATCH updates an existing resource providing only partial data (only some fields).
  - DELETE removes a resource.

OAuth 2.0

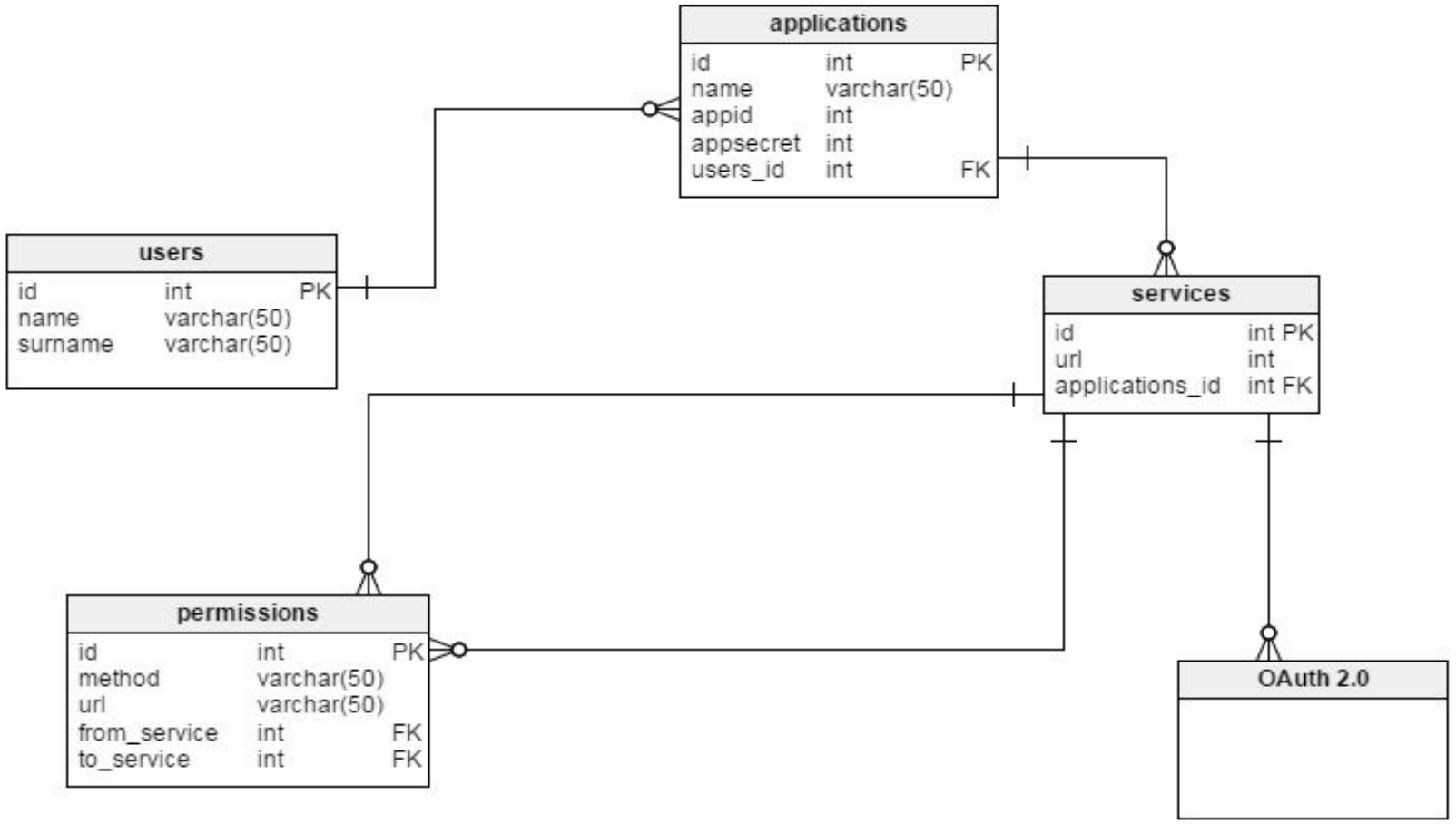
- Next evolution of the OAuth protocol which was originally created in late 2006, enabling applications to access each other's data
- Permissions need to be given to services, not the final user



Proposed Model / Framework



Coordinator E-R Diagram



Application registration to the Coordinator

Application	App. Key	Secret Key	Version	End Point
A	123	ndfg74…	v1	https://registry.domain.edu/v1/student
			v2	https://registry.domain.edu/v2/student
B	234	bchw88…	v1	https://registry.domain.edu/v1/library
C	345	wvfwerf…	v1	https://webservice.domain.edu/v1/bursary/fees
			v2	https://webservice.domain.edu/v2/bursary/fees
			v3	https://webservice.domain.edu/v3/bursary/fees

Roles, list of roles as registered in the Coordinator

App. Key	End Point	Token	Expires
123	https://registry.domain.edu/v1/student	c3fb128c-2571-4133-9b49-643eb134a188	2/15/2016 0:00
234	https://registry.domain.edu/v1/library	dffb128c-2571-4133-9b49-643eb134a188	1/15/2016 0:00
345	https://webservice.domain.edu/bursary/fees	c3fb128c-2571-4133-9b94-643eb134a188	1/15/2016 0:00
478	https://webservice.domain.edu/registry/transcript	6eaf3468-e696-4d0a-958f-f4a24a5efee1	1/15/2016 0:00
563	https://webservice.edu/registry/listofstudents	bd0eb731-171a-40b5-9833-fa5799c0c3ea	10/15/2016 0:00

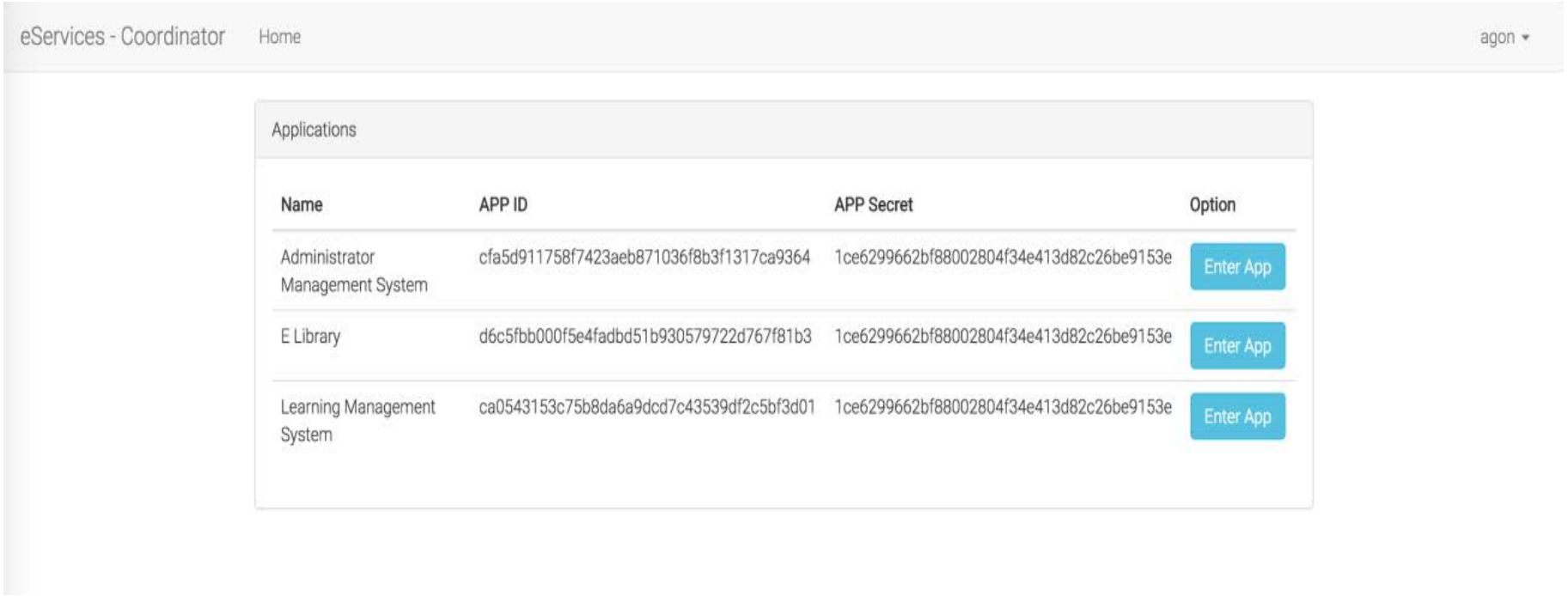
Validation and Case study

- **RESTful Services**
  - **Service 0: Coordinator**
  - Service 1: Administrator Management System
  - Service 2: Learning Management System
  - Service 3: e-Library

- Implemented with open-source PHP web application framework "Laravel" (some of them with version 4.0 and some with version 5.0), MySQL and PostgreSQL as a Database.

Validation and Case study (3)

- Coordinator Interface



Validation and Case study (4)

Permissions for Service **http://lms.agonmemeti.com**

Method	Endpoint	Field	Value	Resource
GET	http://lms.agonmemeti.com/programs/{id}			Print resource
GET	http://lms.agonmemeti.com/courses/{id}			Print resource
POST	http://lms.agonmemeti.com/faculties/store	Field name	Field value	Print resource
DELETE	http://lms.agonmemeti.com/programs/{id}			Print resource
GET	http://lms.agonmemeti.com/faculties/{id}/programs			Print resource

Validation and Case study (5)

Method	Endpoint	Field	Value	Resource
GET	http://elibrary.agonmemeti.com/student/{id}/books			Print resource
POST	http://elibrary.agonmemeti.com/books/store	Field name	Field value	Print resource
PUT	http://elibrary.agonmemeti.com/book/{id}/edit			Print resource

Validation and Case study (6)

Method	Endpoint	Field	Value	Resource
POST	http://lms-usht.agonmemeti.com/addFaculty	Field name	Field value	Print resource
GET	http://lms-usht.agonmemeti.com/list-all-courses			Print resource
GET	http://lms-usht.agonmemeti.com/student-course			Print resource

Validation and Case study (7)

- Administrator Management System

URI	HTTP Method	Collection	Operation	Business Operation
/faculties	GET	faculties	retrieve	Get Faculties
/faculties /create	POST	courses	create	Create new Faculty
/faculties /{faculties_id}/programs	GET	programs	retrieve	Get Study Programs
/programs/{programs_id}	GET	courses	retrieve	Get Program Courses
/courses/{courses_id}	GET	students	retrieve	Get List of Students for specific Program
/courses/{course_id}/edit	PUT	courses	update	Update Program Courses
/faculties/{faculties_id}/programs/{programs_id}	DELETE	programs	delete	Delete Study Program

Validation and Case study (8)

- Learning Management System

URI	HTTP Method	Collection	Operation	Business Operation
/student-get-courses	GET	Courses	retrieve	Get courses
/addFaculty	POST	Faculties	create	Create new Faculty
/student-course	GET	Students	retrieve	Get Students per Courses

Validation and Case study (9)

- E-Library

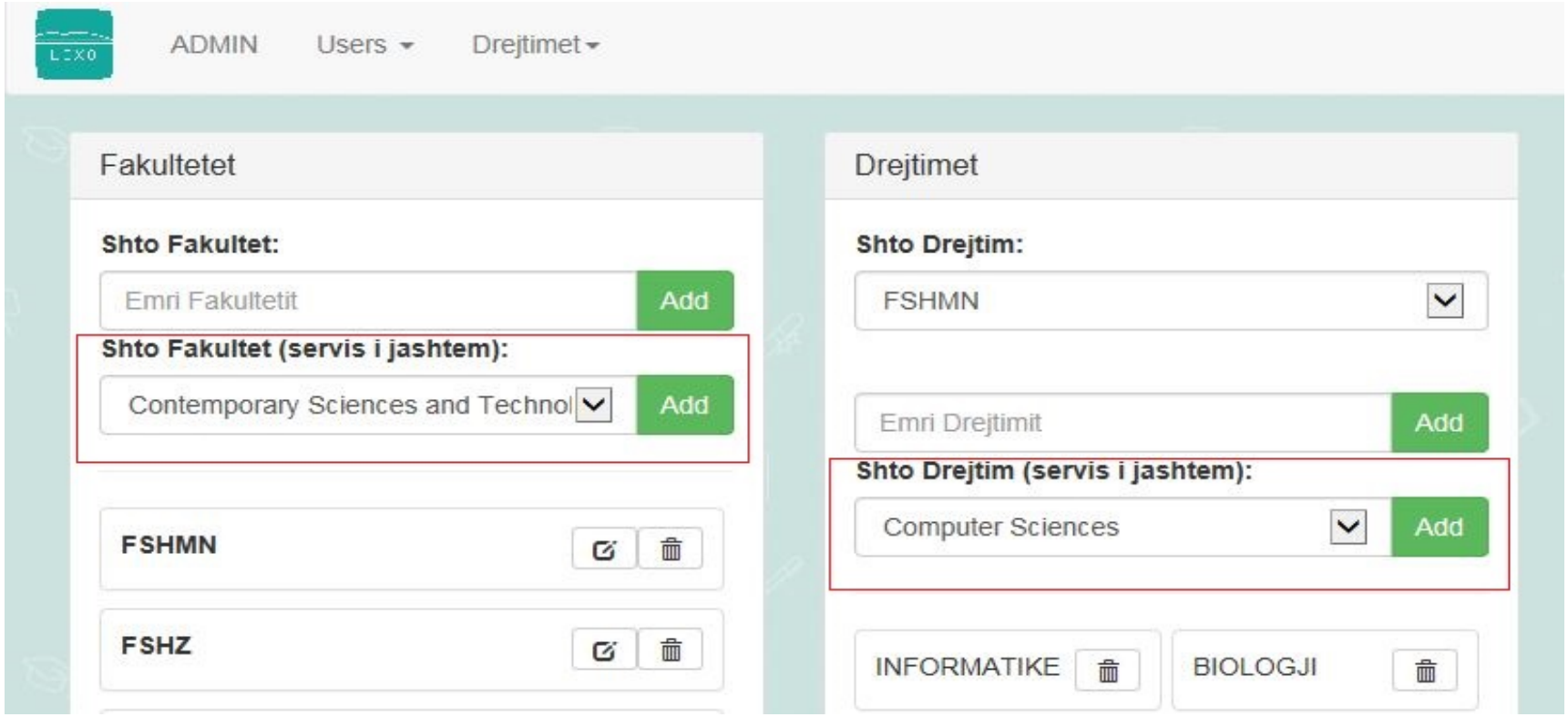
URI	HTTP Method	Collection	Operation	Business Operation
/student/{id}/books	GET	Student books	retrieve	Get Students books
/books/store	POST	books	create	Create new Book
/book/{id}/edit	PUT	books	update	Update Book
/ book/{id}/delete	DELETE	book	delete	Delete a book

Validation and Case study (10)

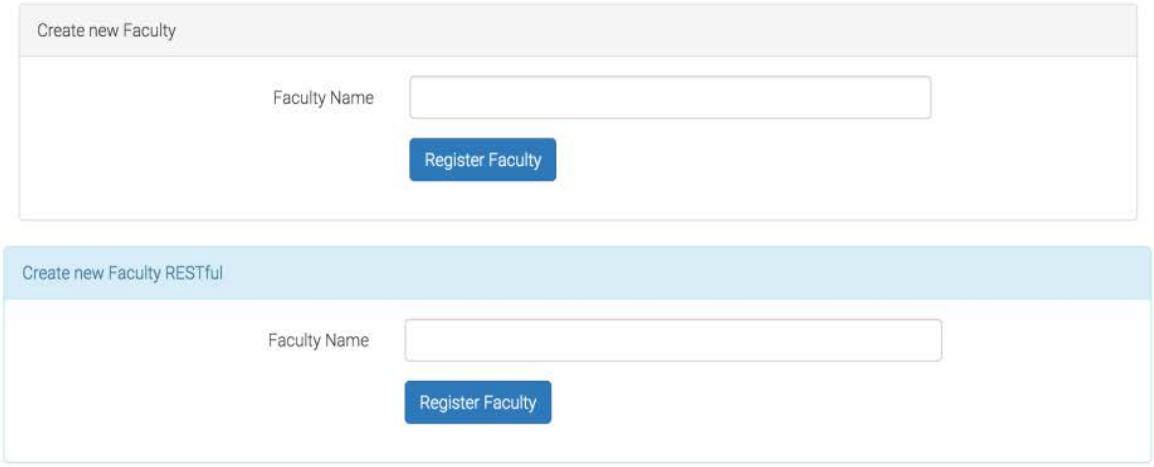
- Service Requests

From_Service	To_Service	HTTP Method	Collection	Operation	Business Operation
B	A	GET	programs	retrieve	Get list of study programs
B	A	POST	faculties	create	Create new Faculty
B	A	GET	faculties	retrieve	Get list of faculties
B	C	GET	books	retrieve	Get list of books
B	A	GET	students	retrieve	Get list of students

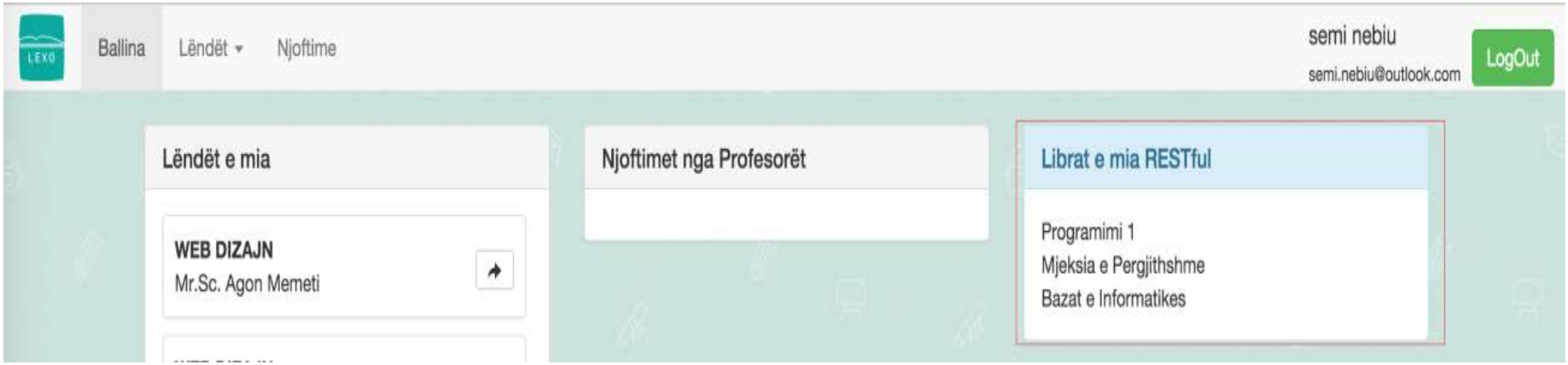
Validation and Case study (11)



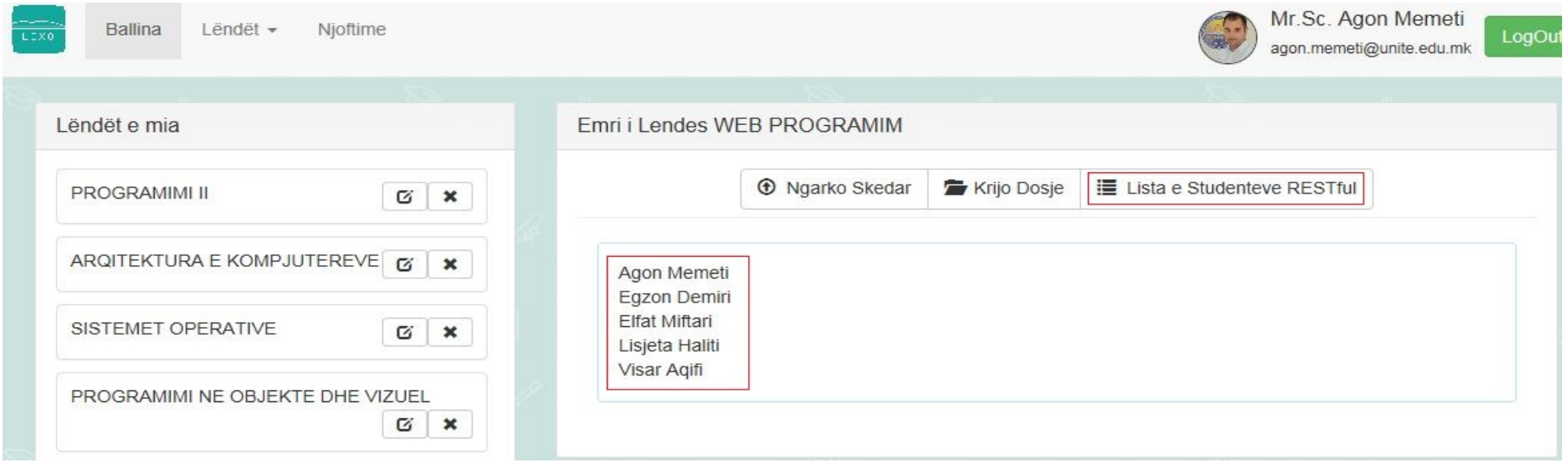
Validation and Case study (12)



Validation and Case study (13)



Validation and Case study (14)



Conclusions

- Defined a model and implemented the framework
  - Facilitate development of in-house services
- Increased flexibility
  - Authorization concerns separated / decoupled from service implementation
  - Developing new services doesn't require changes to existing service/infrastructure
  - Easy to provide testing endpoints
  - Allow the existence of multiple versions of the same service to exist simultaneously
  - Authorization can combine permissions from different services

Future Works

- Framework security issues in detail, which has not been handled and discussed in our case study;
  - Standard security mechanisms can be used for communication
- Integration of the entire University services;
  - Additional criteria should be taken into account;
- Load testing of the coordinator in order to assess the overload limits
- Transfer permissions across service versions
- Integrating workflow solutions with the coordinator

Publications

1. **Agon Memeti** and Betim Cico. Supporting content and learner collaboration and interaction through Cloud computing models. In Proceedings of 6th IEEE International Conference on Computational Intelligence, Communication Systems and Networks, CICSYN 2014**,** pp.145-148, ISBN: 978-1-4799-5075-1, 27-29 May, 2014, Tetovo, Macedonia.

2. Shkumbin Fida, Betim Cico and **Agon Memeti**. Resource Sharing Platform Architecture – Service Oriented Design Lab Environment. In Proceedings of 3rd IEEE Mediterranean Conference on Embedded Computing, MECO 2014**,** pp.285-288, ISBN: 978-1-4799-4827-7, 15-19 June, 2014, Budva, Montenegro.

3. **Agon Memeti**, Dhurate Hyseni and Betim Cico. Cloud computing in Universities with Existing Infrastructure, Case Study: SEEU Research Lab 816. In Proceedings of 5th International Conference "Information Systems and Technology Innovation: projecting trends in New Economy", ISTI 2014**,** pp.16, ISBN: 978-9928-02-471-7, 6-7 June, 2014, Tirana, Albania.

4. **Agon Memeti** and Betim Cico. Building Web Based Applications in the Cloud: A proposed Model, Case Study: Implementation of Several e-services in SEE University in the Cloud. In Proceedings of 9th Annual South East European Doctoral Student Conference, DSC 2014, pp.386-394, ISBN: 978-960-9416-07-8, 25-26 September, 2014, Thessaloniki, Greece.

5. **Agon Memeti**, Besnik Selimi and Betim Cico. Integration of Several University e-Services in the Cloud. In Proceedings of 8th IEEE European Modelling Symposium on Mathematical Modelling and Computer Simulation**,** EMS 2014**,** pp. 360-365, ISBN: 978-1-4799-7412-2, 21-23 October, 2014, Pisa, Italy.

Publications (2)

6. **Agon Memeti**, Besnik Selimi, Adrian Besimi and Betim Cico. A Framework for Flexible REST Services: Decoupling Authorization for Reduce Service Dependency. In Proceedings of 4th IEEE Mediterranean Conference on Embedded Computing, MECO 2015, pp.51-55, ISBN: 978-9-9409-4364-6, 14-18 June, 2015, Budva, Montenegro.

7. **Agon Memeti**, Florinda Imeri and Betim Cico. REST Services Authorization Decoupling through Reusability Approach. In Proceedings of 10th Annual South East European Doctoral Student Conference, DSC 2015, pp.289-297, ISBN: 978-960-9416-08-5, 17-18 September, 2015, Thessaloniki, Greece.

8. **Agon Memeti**, Besnik Selimi, Adrian Besimi and Betim Cico. Coordinating Service Resources: An Architecture for REST Service Collaboration. In Proceedings of 7th International Scientific Conference Computer Science 2015. pp. 280-289, ISBN: 978-619-167-177-9, 08-10 September, 2015, Durres, Albania.

9. **Agon Memeti** and Betim Cico. Learning Management System using REST Services in Cloud Computing. International Journal of Science, Innovation and Technology (IJSINT). Vol.1, No.13, pp. 47-54.Printed ISSN: 2223-2257, Online ISSN: 2225-0751.

10. **Agon Memeti**, Florinda Imeri and Betim Cico. REST Architecture State of Practice in Macedonian IT Companies. Albanian Journal of Natural and Technical Sciences, AJNTS, Vol.20 No.2, pp.97-107, 2015. ISSN: 2074-0867.

Thank You for the Attention!

Questions?