# In-depth Study of RESTful Web Services

School of Computing and Engineering

Khan Najam ul Asre

*Contact e-mail: 21303779@student.uwl.ac.uk*

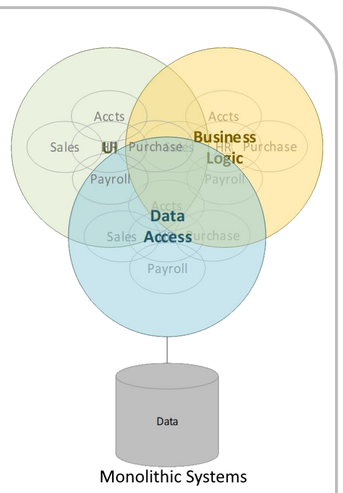
UNIVERSITY OF WEST LONDON

## Abstract

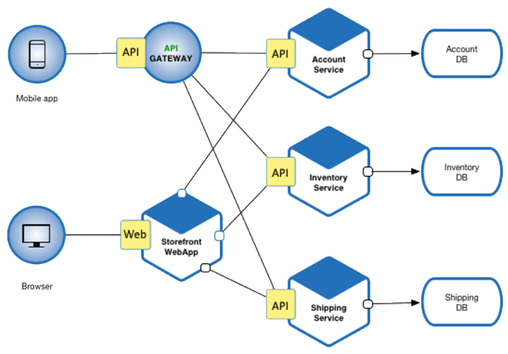
The computing technology has seen enormous development over last few decades. Computing is one of the technologies that is growing at the fastest pace. Computing devices have now become personal pocket items. The devices are versatile, manufactured by many different manufacturers, so have different native platforms and different language tools and frameworks to build application for those devices. This posed many challenges, most notably interoperability, scalability, evolvability, decentralized administration, efficiency and performance. To meet such challenges there is a need for an architectural style that is platform and language agnostic and capable of achieving the attributes of modern systems.
**Representational State Transfer (REST)** is the most recognised architectural style that offers all this.
The project involved extensive study of literature on REST both online and in-book, most notably the PhD dissertation **"Architectural Styles and the Design of Network-based Software Architectures"** by **Roy Thomas Fielding** in which he introduced REST in detail. The project involved study of theory and then learning the implementation

## Introduction

Early computers had limited capabilities, despite, they were too expensive and too large in size for individuals to. Hence, they were used only within the large organisations. The software systems were **monolithic** where functionally distinct concepts, e.g. data persistence and retrieval, business logic, user interface, etc. were strongly interwoven without any clear boundaries or architectural separation. Business requirements would change rarely and hence the software systems. A version of software would remained useful, so monolithic applications worked really well.
Then came the Internet and it provided a global network infrastructure. This gave rise to the **distributed systems**. Monolithic architecture did not suit distributed systems. There was a need for architecture which could support the distributed components running in various platforms to interoperate, evolve independently without the need of centralized administration. This led to the popularity of Microservices Architecture. Advent of Cloud required scalable and interoperable systems. **REST** best fitted such requirements and gained significant popularity. Most notable organisations have migrated to REST. Therefore it is important to understand REST in depth, and this project is carried out to gain such understanding




Monolithic Systems




*Microservices Architecture*

## Aims and Objectives

### Aims

The aim of this project is to study and understand in greater detail the concept of RESTful architectural pattern of designing web services, thereby evaluating the RESTful services compared to other technologies like SOAP (Simple Object Access Protocol) services and RPC (Remote Procedure Call).

### Objectives

1. To carry out in depth study of REST principles in order to understand how RESTful services differ from other options.
2. To build a data intensive web API software using REST architectural style to help understanding and evaluation.
3. To gain a detailed understanding of the nature of HTTP protocol and how HTTP and RESTful architectures are related to each other.
4. To make use of at least one HTTP client testing tool to gain a practical experience of how HTTP (and thereby RESTful) communication works.
5. To understand the software project life cycle.

## Conclusions

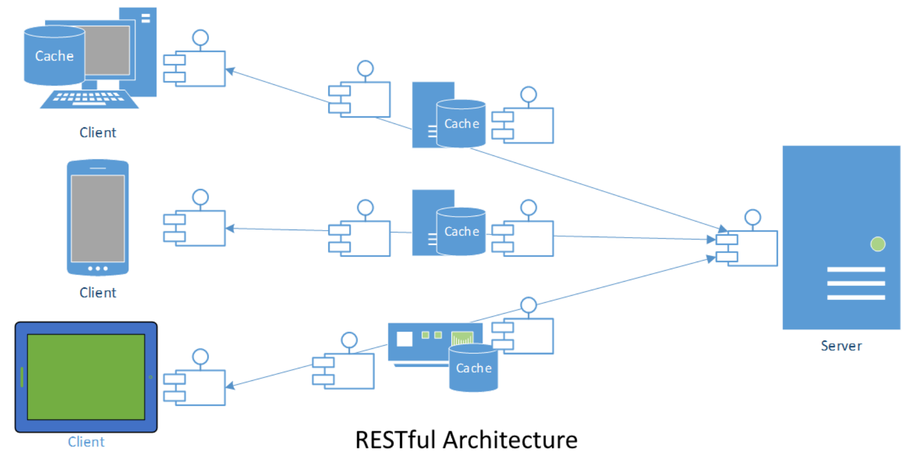
Today's abundance of versatile devices, easy access to the Internet and frequently and quickly changing business requirements demand for the systems that elegantly address the required system attributes like interoperability, scalability, visibility, reliability, performance, efficiency, decentralised administration, eveolvability, continuous delivery and integration. We learned that more or less these were the attributes that were the key drivers behind the development of HTTP Standards that is driving the global Internet today. The key designer of the HTTP and URI standards, Roy Thomas Fielding, summarised his nearly a decade's work in his PhD dissertation and named the style behind the design of HTTP and Web as Representational State Transfer (REST). We learned that the REST is the best architectural style for the development of modern network bases systems. We also learnt about a number of misconceptions about the REST e.g. relationship of REST with HTTP, JSON, SOAP etc. We learnt that REST is style, not a technology, standard, specification or design pattern – A style composed of concepts that adhere to certain constraints and exhibit certain behaviour. We concluded that REST is best suited style of architecting demanding applications of today's versatile world of computing and smart devices.

## References

Fielding, R. (2000). UNIVERSITY OF CALIFORNIA, IRVINE. PhD. UNIVERSITY OF CALIFORNIA, IRVINE.
Fielding, et al. (1999) IETF, HTTP/1.1, RFC2616

## Methodology

A strategy was devised which involved detailed study of online and in-book literature as well as reading through the fundamental Roy Fielding's PhD dissertation to gain theoretical knowledge. Then to apply such knowledge to implement a reasonably sized application built upon RESTful architectural style using industry popular Microsoft ASP.NET Core, .Net Core Framework, Microsoft Visual Studio and Microsoft SQL Server relational database engine.




RESTful Architecture

Above picture is generic high-level illustration of RESTful systems that was perceived thought the detailed study of the literature and practical experience gained through implementation of REST concepts by developing a social task management application named **TaskBook**. This offered the level of understanding required to make comparison of REST with SOAP and RPC styles and to understand how REST is a better style for architecture modern applications through critical evaluation.

## Implementation

The implementation began with the specifying requirement for the **TaskBook**. The next step was to identify the **entities** involved and **relationship** between those entities. The main focus remained on the core implementation of REST concept rather than overhead of boilerplate code. To help this the selection of tools and technology was made very carefully. The result was the selection of tools that offered almost all of boilerplate stuff, e.g. URL routing (ASP.NET MVC), Authentication and Authorisation (Microsoft Identity), out-of-the-box. This allowed the maximum concentration on the project topic, the REST.

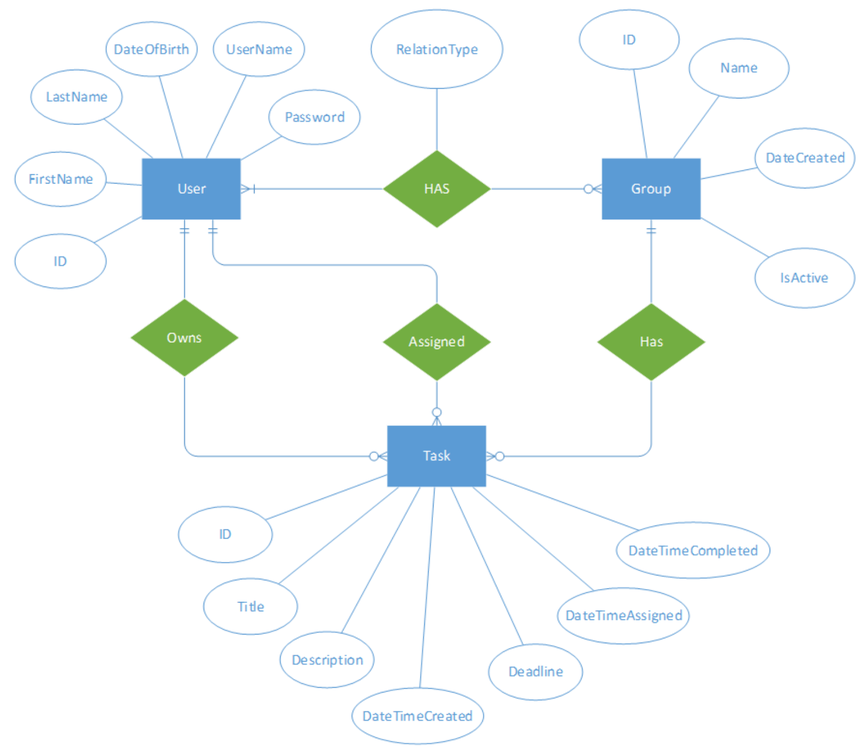



*Figure 4-1 Domain Model*

Following **Resources** were identified along with supported **HTTP Methods**, and were then implemented by **mapping** these resources to above **entities**.

| | |
|---|---|
| **Accounts** (GET,HEAD,POST,OPTIONS) | **TaskAssignments** GET,OPITONS,HEAD,PUT,DELETE) |
| **Credentials** (PUT, OPTIONS) | **TaskCompletions** (GET,OPTIONS,HEAD,PUT,DELETE) |
| **GroupMemberships** (GET,DELETE,HEAD,POST,OPTIONS) | **UserGroups** (GET,POST,OPTIONS,HEAD,PUT,DELETE) |
| **GroupTasks** (GET,POST,OPTIONS,HEAD,PUT,DELETE) | **UserMemberships** (GET,OPTIONS,HEAD) |
| **Profiles** (GET,PUT,OPTIONS,HEAD,PATCH) | **UserTasks** (GET,OPTIONS,HEAD) |