

Best Practices in Implementing RESTful Web Services

Nathan Hughes - 1809708

May 8, 2019

Abstract

This report discusses the best practices for implementing a RESTful web service, providing examples from a project developed by the author illustrating these principles.

1 Introduction

Representational State Transfer (REST) was introduced by Roy Fielding in his PhD dissertation published in 2000 [1]. In this thesis, Fielding describes REST as an architectural style that has dictated the development of the Web since HTTP 1.0 was introduced in 1996.

The adoption REST for developing web services can be contrasted with the *Simple Object Access Protocol* (SOAP), designed in 1998 by Winer, Box, Atkinson and Al-Ghosein [2]. SOAP consists of well-defined and official practices when being used to implement a web service [3]. No such official standards exist for REST.

This report discusses the best practices for implementing RESTful web services as defined by the literature. Concurrently, example code is given from the accompanying auction site project to demonstrate RESTfulness. A discussion of the wider adherence to RESTful principles is also given.

2 Principals of REST

RESTful services are primarily concerned with three concepts, *resources*, *verbs* and *representations* [4, 5, 6, 7]. The resources of a REST service are data that the service wishes to supply explicitly. Examples from the accompanying auction site include **batches**, **bids** and **users**. The representation of a resource is simply the format in which the resource is displayed - JSON, for example. Verbs are methods that a client can use to access the resources of the web service. Resources are identified via a *Universal Resource Identifier* (URI), and it is from this URI the representation of a resource is obtained.

An example unifying these topics is the **batches** resource in the auction site. It can be access via the URL `http://localhost:8080/batches` and is represented in JSON format. Verbs such as **GET** and **POST** can be used to obtain this representation.

RESTful services are also defined by 6 different architectural constraints [1, 5, 8]. These were identified by Fielding as phenomena that, when followed, aided the scalability of the Web.

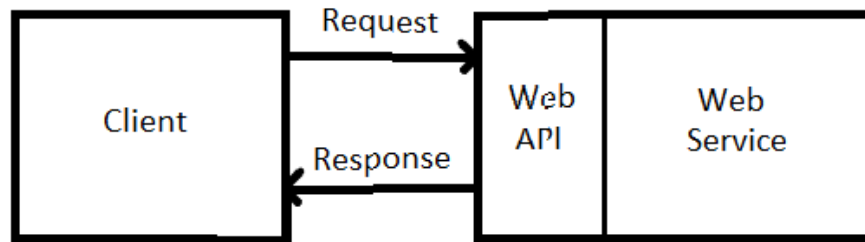


Figure 1: Abstract illustration of client-server model, adapted from [5].

2.1 Client-Server

Client-server architectures involve two agents between which data is transferred [9]. The client sends requests to the server, to which the server will respond. The additional structure present in a REST service is that services are exposed via a Web API [5]. There is separation of concerns between these agents; client-side logic is handled only by the client and server-side logic in the server. The client in the case of the auction site is the browser used to view the HTML pages. JavaScript is used to send requests to the URIs of various resources provided by the server back-end.

2.2 Uniform Interface

The success of a agent interfacing with the Web is heavily dependent of the adherence to existing standards. Clients will need to send requests that the servers can interpret, and the servers in turn will need to respond with data that is intelligible to the client. Fielding identified four key properties of a uniform interface - the existence of a URI, clients manipulate resources via representations, messages give meaningful descriptions of themselves and representations containing links to related representations (HATEOAS) [1, 5]. In the auction site example, the client uses the standard **GET** and **POST** verbs to request and send **application/json** messages, and the server responds with data of the same representation.

2.3 Statelessness

A RESTful service is stateless when the server does not record the state of its clients, and therefore the client must manage this information [5, 8]. The server for the auction project does not record any data from any client's application. Instead, the client handles any state-dependant tasks. In the auction site, the representations at each API are independent of the state of any client utilising the information.

2.4 Cacheability

The cacheable restriction requires that each response sent by a server should be assigned a cacheability. Responses that are cached are stored away from the server, closer to the client, which increases the speed of using the service for the clients and reduces the volume of requests sent to the server [5, 8]. The AJAX methods used to retrieve data are set to cacheable by default [10].

2.5 Layered System

The client and server in the REST service separate their concerns so that there is no overlap in functionality. The layered system criterion adds an additional restriction in that the messages sent by either the client or the server are agnostic to the recipient [5]. This allows the routing of messages to authenticators prior to reaching the intended destination, for example.

2.6 Code on Demand

Code on Demand describes any situation in which a web server will transfer code to the client for temporary use [5]. The auction site has no need to provide any code to the client.

3 Architectural Practices for RESTful Web Services

The restrictions specified by Fielding provide a rough guide as to the structure of a RESTful web service. The finer details of the implementations were discussed in later publications based on the experiences of the authors and scientific analysis of existing services [5, 7, 11]. A selection of these findings are discussed in this section.

3.1 Structure of URIs

The structure of the path in a URI, according to Masse, should consist of separating slash symbols (/) to portray a hierarchy, with no slash symbol at the end [5, 7]. The resources exposed by the API should be named with nouns [4]. Masse and Palma *et al.* also recommend that the nouns in the URI should be pluralised when describing resources [5, 12]. Additionally, URI should not contain any CRUD (Create, Read, Update, Delete) verbs, and query parameters should also be free of any said verbs [11, 4]. These query components should be used to filter collections [5, 12]. Following these patterns lessens confusion for users navigating the API [12].

```
▼ bids:
  href: "http://localhost:8080/bids?page,size,sort"
  templated: true
▼ localoffices:
  href: "http://localhost:8080/localoffices?page,size,sort"
  templated: true
▼ users:
  href: "http://localhost:8080/users?page,size,sort"
  templated: true
▼ plates:
  href: "http://localhost:8080/plates?page,size,sort"
  templated: true
▼ profaneplates:
  href: "http://localhost:8080/profaneplates?page,size,sort"
  templated: true
▼ profile:
  href: "http://localhost:8080/profile"
```

Figure 2: Plural nouns for resources.

```
▼ 18:
  batchid: 1832
  batchprefix: "K219 EV"
  enddate: "2019-05-21T00:57:27.000+0000"
  availability: true
  price: 360000
  _links:
    self:
      href: "http://localhost:8080/batches/1832"
    batch:
      href: "http://localhost:8080/batches/1832"
    plates:
      href: "http://localhost:8080/batches/1832/plates"
    bids:
      href: "http://localhost:8080/batches/1832/bids"
```

Figure 3: Example separating slashes and HATEOAS.

It is also recommended that the Web API be defined as part of the sub-domain, defined as `http://api.examplewebsite.com` rather than `http://examplewebsite.com/api` [11]. This is to facilitate the consumption of the service by multiple clients.

3.2 Request Methods

The desirable properties of the HTTP methods were originally defined by Fielding [1]. Richardson and Ruby, Vinoski and Masse all suggest similar usages of the primary HTTP methods [7, 6, 5]. They suggest that the **GET** method should be used to retrieve representations from resources, the **POST** method is to be used only to create new resources to a resource and the **PUT** method should be used to post or update resources to a specific URI. These are further justified by Fielding and Reschke [13]. In the particular case of the **POST** method, Richardson and Ruby advise against *overloading* - using the **POST** method to specify any other non-HTTP methods - since this behaviour is inherited from Remote Procedure Call services, which are not RESTful [7].

```
$.ajax({
  type: 'POST',
  contentType: 'application/json',
  url: 'http://localhost:8080/users',
  crossDomain: true,
  data: JSON.stringify(data),
  dataType: 'json',
  timeout: 600000,
  success: console.log("posted" + JSON.stringify(data)),
  error: function (err) {
    if (err.readyState === 0) {
      window.alert("This email is already registered");
    }
  }
});
```

Figure 4: AJAX with POST used to create a new account with unspecified URI.

```
batch.price = bidvalue;
$.ajax({
  url: "http://localhost:8080/batches/" + batchid,
  type: 'PUT',
  contentType: 'application/json',
  crossDomain: true,
  data: JSON.stringify(batch),
  success: console.log("updated batch"),
  error: function(err) {
    console.log("Error updating batch: " + String(err));
  }
});
```

Figure 5: AJAX with PUT used to update a batch using specific URI.

3.3 Error Handling

The consensus reached by the literature is that REST services should use a subset of the HTTP statuses to handle the state of their service, which should be represented as a JSON message [5, 14]. In particular, the 200 status should be used to indicate a success and no errors should be contained in its response body. Multiple 4xx error codes are recommended, such as 401 for unauthorized access attempt and 404 when the URI does not map to an existing resource.

3.4 Structure of Metadata

The header of any message should preserve the intended structure of the associated HTTP method [5]. In particular, this preserves the desired idempotent property of the **GET**, **PUT** and **DELETE** methods [6]. The header of each message send must specify the content type of the content [5, 6]. Examples of specifying content type can be seen in figures 4 and 5.

3.5 Structure of Content

As mentioned by Fielding, HATEOAS is a heavily desired property of a RESTful web service [1]. That is, when a resource uses data that is also represented by another resource, it is good practice to use links to refer to this data. In particular, it is advised that a representation contains a *self-link*. An example of this is available in figure 3.

4 Non-RESTfulness of Well-Known Web Services

Despite best practices for RESTful web services being a readily available resource, many well-known public APIs fail to meet the standards set out by Fielding and subsequent authors [15, 16, 17, 11, 18].

Work published by Bülthoff and Maleshkova in 2014 analysed 45 Web APIs, including Amazon S3, Azure, GitHub, Twitter and Wikipedia, and concluded that REST best practice adoption was poor [15]. For example, less than 20% of APIs analysed used HATEOAS principles in designing their resources, suggesting that the low rate of adoption is due to its difference from previous web services principles, such as Remote Procedure Call.

Rodriguez, Baez and Daniel conducted an empirical analysis of HTTP traffic collected by an internet provider to determine the adoption of REST principles in 2016 [11]. Their study used the Richardson Maturity Model to analyse the structure of APIs and their compliance to REST practices [19]. They found that about 10% of HTTP requests used tunneling - wrapping an operation in a GET or POST request, whereas 90% did not adopt HATEOAS.

A more recent 2018 study by Neumann, Laranjeiro and Bernardino analysed the RESTfulness of 500 websites listed in the `Alexa.com` top 4000 website rankings claiming to provide RESTful APIs [17]. They found that just 4 of said APIs matched their criteria for RESTfulness. Additionally, only 21 used HATEOAS principles, and only 5 provided a uniform interface.

Overall, the adoption of common REST guidelines and best practices is low and adoption by newer APIs appears to be stagnant. It is possible that this stagnation is explained by the familiarity effect - if the majority of APIs that developers are exposed to tend not to follow these principles, it is plausible to predict that APIs designed by these developers will continue to reject said principles, although no formal studies have been conducted to investigate this.

5 Conclusion

The guidelines for implementing a RESTful web service and API are expansive and well-documented. At least some of the original restrictions set out by Fielding have been applied in some well-known public APIs. Further theoretical work has been conducted into the best practices for specific components of a RESTful API implementation. These practices aid the user in understanding the structure due to the uniformity achieved and reduce the non-compliance with Fielding's original restrictions. Despite this, the wider implementation of these principles is lacking, as demonstrated by multiple studies.

References

- [1] R. T. Fielding, “REST: architectural styles and the design of network-based software architectures,” Doctoral Dissertation, University of California, Irvine, 2000.
- [2] D. Ferguson, “Exclusive .net developer’s journal ”indigo” interview with microsoft’s don box,” *.NET Developer’s Journal*, August 2004, accessed: 08/05/2019. [Online]. Available: <http://dotnet.sys-con.com/node/45908>
- [3] M. Gudgin *et al.* (2007) Accessed: 08/05/2019. [Online]. Available: <https://www.w3.org/TR/soap12-part1/>
- [4] D. Kumar, “Best practices for building RESTful web services,” Infosys, Tech. Rep., 2018.
- [5] M. Masse, *REST API Design Rulebook*, 1st ed. O’Reilly Media, 10 2011.
- [6] S. Vinoski, “Restful web services development checklist,” *IEEE Internet Computing*, vol. 12, no. 6, pp. 96–95, Nov 2008.
- [7] L. Richardson and S. Ruby, *Restful Web Services*, 1st ed. O’Reilly, 2007.
- [8] C. Pautasso, “RESTful web services: Principles, patterns, emerging technologies.” Springer, 2014, pp. 31–51.
- [9] A. Berson, *Client/Server Architecture*. New York, NY, USA: McGraw-Hill, Inc., 1992.
- [10] JS Foundation. Documentation - jQuery.ajax(). Accessed: 08/05/2019. [Online]. Available: <http://api.jquery.com/jquery.ajax/>
- [11] C. Rodriguez, M. Baez, F. Daniel, F. Casati, J. Carlos Trabucco, L. Canali, and G. Percannella, “Rest apis: A large-scale analysis of compliance with principles and best practices,” 06 2016, pp. 21–39.
- [12] F. Palma, J. Gonzalez-Huerta, M. Founi, N. Moha, G. Tremblay, and Y.-G. Guhneuc, “Semantic analysis of restful apis for the detection of linguistic patterns and antipatterns,” *International Journal of Cooperative Information Systems*, p. 1742001, 05 2017.
- [13] R. Fielding and J. Reschke. (2014) Hypertext transfer protocol (HTTP/1.1): Semantics and content. Accessed: 07/05/2019. [Online]. Available: <https://www.ietf.org/rfc/rfc7231.txt>
- [14] M. Stowe, “Undisturbed REST: A guide to designing the perfect API,” 2015.
- [15] F. Bülthoff and M. Maleshkova, “RESTful or RESTless - current state of today’s top web APIs,” in *The Semantic Web*, 2014, pp. 64–74.
- [16] J. Kopeck, P. Fremantle, and R. Boakes, “A history and future of web APIs,” *it - Information Technology*, vol. 56, 01 2014.
- [17] A. Neumann, N. Laranjeiro, and J. Bernardino, “An analysis of public rest web service apis,” *IEEE Transactions on Services Computing*, 2018.
- [18] F. Petrillo, P. Merle, N. Moha, and Y. G. Guéhéneuc, “Are REST APIs for cloud computing well-designed? an expolatory study,” *Service-Oriented Computing*, pp. 157–170, 2016.

- [19] M. Fowler. (2010, March) Richardson maturity model. Accessed: 08/05/2019. [Online]. Available: <https://martinfowler.com/articles/richardsonMaturityModel.html>

Serverless Technologies and the Circuit Breaker Pattern

Nathan Hughes - 1809708

May 8, 2019

Abstract

This report discusses available cloud services and their application to hosting web services. This is followed by a discussion of the Circuit Breaker pattern when applied to improving the resiliency of serverless web services.

1 Introduction

Mell and Grance define cloud computing as a model that provides on-demand access to a collection of computing resources, configurable by the user with quick deployment times with little management by the service provider [8]. That is, cloud providers, such as MicroSoft Azure and Amazon Web Services, provide clients access to a part of their array of servers that can be configured by the clients themselves.

The service provided by these providers can vary dramatically in utility. These options vary not only by the amount of computing power or memory provided, but by scope of possible service and abstraction. Three standard service models are the Infrastructure as a Service, Platform as a Service and Software as a Service [8]. Infrastructure as a Service, for example, will provide virtualised computer infrastructure components in lieu of an on-premises infrastructure such as storage and hardware, whereas Software as a Service will provide clients with software that is hosted on the cloud for their consumption.

Prior to the advent of readily-available clouds, development for companies was prone to being expensive, since hardware and servers was locally required [11]. This rendered the smallest companies unable to compete with well-financed enterprises. Cloud computing facilitates the operations these smaller companies [12].

This report will explore the benefits of using serverless products generally and also in the specific case of web services. The Circuit Breaker pattern is then discussed in the context of improving the resiliency of these web services.

2 Serverless Computing

Serverless computing is a paradigm for executing software where the provider of the cloud manages the allocation of servers on which the runtime takes place [6]. The allocation of servers is elastically administered to handle requests. Serverless services are also known as Function as a Service.

Serverless frameworks have found applications in fields such as image processing and computing in scientific research [2]. Of particular note is the application of serverless technologies to hosting web applications. A simple client-server application, for example, will keep all non-client functionality in a server located with the server provider [1]. Serverless computing allows the developers to relocate some of this functionality to a serverless computing environment such as authentication services or Business Intelligence systems [10].

Using serverless technologies for hosting web services provides unique benefits. When a web service experiences inconsistent traffic, a local server will need to be able to accommodate the maximum volume of traffic. As such, the cost of the hardware will always be proportional to the maximum volume of traffic. This is contrasted with serverless web services - the elastic deployment of the providers resources not only allows for varying amounts of computing power, reducing the cost for the user, paying only for the resources used, but also reduces the amount of energy used in sustaining the service [10].

Serverless services do produce challenges and difficulties not previously encountered with on-premises server architectures. The management of said services can be limited, if not completely out of the developers control [6]. Consequently, developers are unable to affect the state of their applications and must wait for the provider to correct their service. A recent example of this is the Microsoft outage which disabled user's access to services such as Azure, OneDrive and Office 365 Sharepoint [3].

Further problems originate from the state of data stored in the cloud. As Roberts explains, whilst data storage is available, there is no possible way to guarantee that the state persist between different runs of a function [10]. As such, it is recommended that any such data is stored externally to the serverless function.

3 Circuit Breaker Pattern

Services hosted in the cloud usually access external and remote resources in order to carry out its intended function, such as a persistent storage in the form of a database. If problems occur while trying to connect to these remote resources while the service is running, it is possible for the service to account for the downtime through use of a Retry pattern [4]. With this pattern, anticipated failures, such as temporary database downtime due to a high load of traffic, can be accounted for by waiting a short amount of time before retrying the queued operation. In the case of unexpected or long-term issues, such as the disabling of a server, clients could be made to wait for extended periods of time before a fault is reported back [4]. The *circuit breaker* pattern can be employed to account for the error, replying to client requests with a fault message while tripped [9]. The circuit breaker pattern is a tool used to bolster the resiliency of a system utilising cloud technology [9].

3.1 States of the Circuit Breaker

The circuit breaker pattern can exist in three different states, *closed*, *open* and *half-open*. Each is visited in succession when a possible fault is being detected. An accompanying `faultCounter`

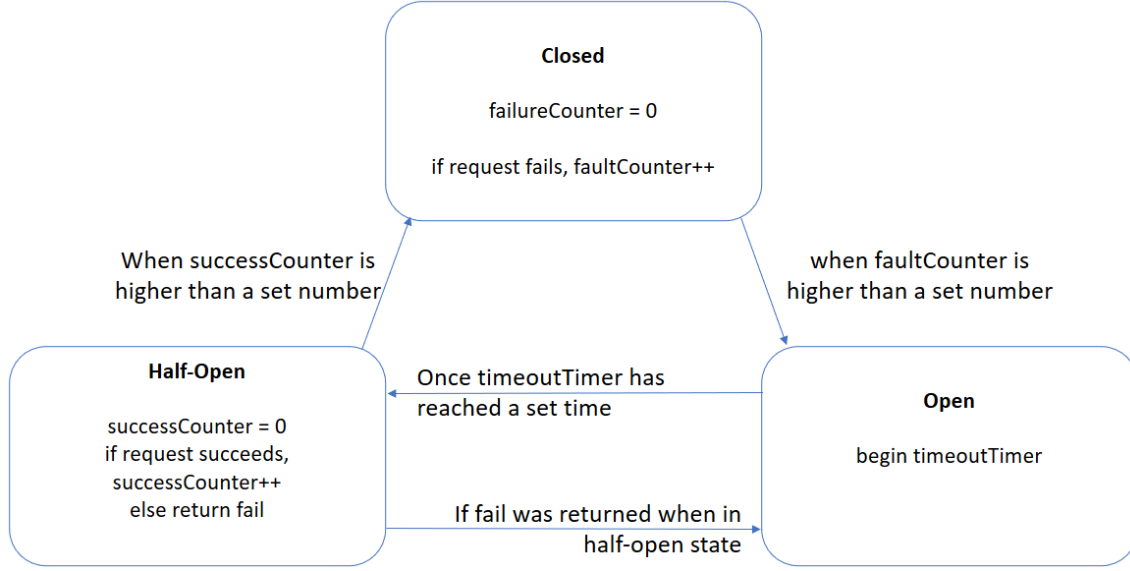


Figure 1: Illustration of the flow between states, adapted from [4].

describes the amount of faults experienced when attempting to communicate with the service attached to the circuit breaker, and a `successCounter` that counts the number of successes. A `timeoutTimer` is also present for use when the circuit is open. The synthesis of these objects is explained below. The explanation below is adapted from documentation provided by Microsoft and a paper by Montesi and Weber [4, 9].

3.1.1 Closed

The Closed state is the state that the circuit exists in when no faults are being experienced. When experiencing a faulty request, the circuit breaker tracks this fault by incrementing `failureCounter`. This value is set to 0 after a period of time to ensure that `failureCounter` is only sensitive to high intensity of faulty requests. In the event that the multiple faults are detected, more than a pre-determined threshold, the circuit breaks, and is now in the Open state.

3.1.2 Open

Once the circuit is set to the Open state, the `timeoutTimer` begins. No requests are sent to the service until this timer has finished counting down. Upon finishing this timer, the circuit is converted to the Half-Open state.

3.1.3 Half-Open

Upon reaching the Half-Open state, a requests are allowed to be sent to the service once again. The caveat is that the successes of these request are now being recorded by the `successCounter`. When the `successCounter` reaches another pre-determined value, the circuit is put into the Closed

state. If any of these requests fail, the circuit is switched back to the Open state. The circuit breaker is not responsible for the recovery effort following a break in the circuit. This is usually handled externally, usually developers restarting components [7].

3.2 Variations of the Circuit Breaker

Whilst the general implementation of the circuit breaker patterns does not vary greatly, its location in the architecture affects the outcome of a trip in the circuit. There are three main locations for a circuit breaker to be implemented in a cloud service, either on the client's side, the service's side or as a proxy [9].

3.2.1 Client-side Circuit Breaker

Circuit breakers present on the client side of the architecture will monitor the connection between the client and the remote service or resource. This connection is tripped when enough errors are detected in attempt to *fail fast* - assuming the worst-case scenario that the service is no longer responsive - preventing any subsequent calls to the service [9]. The disadvantage to circuit breakers located on the client's side of the architecture is that it assumes clients can be made to use the circuit breaker and that it won't be used maliciously [9].

3.2.2 Service-side Circuit Breaker

Requests sent by clients to a service will first be passed to the service-side circuit breaker, which dictates if the requests is to be carried out, according to the state of the circuit. This removes the need to make any assumptions about the client's use of the circuit breaker, but now requires that the circuit is used for every single request independently of the circuit's current state [9].

3.2.3 Proxy Circuit Breaker

To alleviate the problems with the invasive nature of both client- and service-side circuit breakers, it is recommended that the circuit breaker is deployed as a proxy between them [9, 7]. The proxy will contain a circuit breaker for each client and service, and requests are sent from a client to a server only if both of their respective circuits are in a closed state. The benefit of this method is that it facilitates the sharing of data between the circuit breakers; when multiple client circuit breakers are switched to the open state, for example, it can inform the rest to open if the evidence of fatal system failure is strong enough [9].

4 Conclusion

The recent development of widely-available and inexpensive cloud resources has facilitated the departure from expensive on-location server architectures, for example [12, 5]. This, in turn allows more enterprises to take advantage of server-based architectures previously unavailable to them. Cloud-based architectures also experience a greater degree of elasticity and scalability due to the readily available computing potential made available to clients by cloud providers [12, 5].

This report discussed the specific technology of serverless services, or Function as a Service.

Serverless technologies create new problems in system resiliency. One possible design pattern intended to strengthen a system's resiliency is the Circuit Breaker pattern. Once implemented as a

proxy, this object will manage the connections to both the client and the server and make decisions on which requests are allowed to be executed based on the state of the system.

References

- [1] A. Berson, *Client/Server Architecture*. New York, NY, USA: McGraw-Hill, Inc., 1992.
- [2] G. C. Fox, V. Ishakian, V. Muthusame, and A. Slominski, “Status of serverless computing and function-as-a-service(faas) in industry and research,” 2017.
- [3] C. Harbison, “Microsoft outage: Azure, office 365 sharepoint, onedrive and more down leaving users unable to access features,” *Newsweek*, May 2019.
- [4] A. Homer, J. Sharp, L. Brader, M. Narumoto, and T. Swanson, *Cloud design patterns - prescriptive architecture guidance for cloud applications*. Microsoft, 2014.
- [5] E. Jonas, Q. Pu, S. Venkataraman, I. Stoica, and B. Recht, “Occupy the cloud: Distributed computing for the 99%,” in *Proceedings of the 2017 Symposium on Cloud Computing*, ser. SoCC ’17, 2017, pp. 445–451.
- [6] K. Kritikos and P. Skrzypek, “A review of serverless frameworks,” in *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, Dec 2018, pp. 161–168.
- [7] V. Malcher, “Design patterns in cloud computing,” in *2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, Nov 2015, pp. 32–35.
- [8] P. Mell and T. Grance. (2009) The nist definition of cloud computing. National Institute of Standards and Technology. Accessed 08/05/2019. [Online]. Available: <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf>
- [9] F. Montesi and J. Weber, “From the decorator pattern to circuit breakers in microservices,” in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, ser. SAC ’18, 2018, pp. 1733–1735.
- [10] M. Roberts. (2018) Serverless architectures. Accessed: 08/05/2019. [Online]. Available: <https://martinfowler.com/articles/serverless.html#unpacking-faas>
- [11] J. Voas and J. Zhang, “Cloud computing: New wine or just a new bottle?” pp. 15–17, 2009.
- [12] H. Wang, W. He, and F.-K. Wang, “Enterprise cloud service architectures,” *Information Technology and Management*, vol. 13, 12 2012.

Plate.java

```
package com.auction.model;

public abstract class Plate {

    int plateid;
    String numberplate;

    public Plate() {
    }

    public Plate(int plateid, String numberplate) {
        this.plateid = plateid;
        this.numberplate = numberplate;
    }

    public int getPlateid() {
        return plateid;
    }

    public void setPlateid(int plateid) {
        this.plateid = plateid;
    }

    public String getNumberplate() {
        return numberplate;
    }

    public void setNumberplate(String numberplate) {
        this.numberplate = numberplate;
    }

}
```

NumberPlate.java

```
package com.auction.model;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import javax.persistence.*;

@Entity
@Table(name="plates")
@JsonIgnoreProperties({"hibernateLazyInitilaizer", "handler"})
public class NumberPlate extends Plate implements java.io.Serializable {

    // Attributes

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int plateid;
```

```

private String numberplate;

@ManyToOne(fetch = FetchType.LAZY)
@JoinColumn(name = "batchid")
private Batch batch;

// Constructors

public NumberPlate() {

}

public NumberPlate(int plateid, String numberplate) {
    super(plateid, numberplate);
}

public NumberPlate(String numberplate, Batch batch) {
    this.numberplate = numberplate;
    this.batch = batch;
}

public NumberPlate(int plateid, String numberplate, Batch batch) {
    super(plateid, numberplate);
    this.batch = batch;
}

// Properties

public int getPlateid() {
    return plateid;
}

public void setPlateid(int plateid) {
    this.plateid = plateid;
}

public String getNumberplate() {
    return numberplate;
}

public void setNumberplate(String numberplate) {
    this.numberplate = numberplate;
}

public Batch getBatch() {
    return batch;
}

public void setBatch(Batch batch) {
    this.batch = batch;
}

// Methods
}

```

Batch.java

```
package com.auction.model;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import javax.persistence.*;
import java.sql.Timestamp;
import java.util.List;

@Entity
@Table(name = "batches")
@JsonIgnoreProperties({"hibernateLazyInitilaizer", "handler"})
public class Batch implements java.io.Serializable {

    // Attributes

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name="batchid")
    private int batchid;

    private String batchprefix;

    private Timestamp enddate;

    @OneToMany(fetch = FetchType.LAZY, targetEntity = NumberPlate.class, mappedBy = "batch")
    private List<NumberPlate> plates;

    private boolean availability;

    private int price;

    @OneToMany(fetch = FetchType.LAZY, targetEntity = Bid.class, mappedBy = "batch")
    private List<Bid> bids;

    // Constructors

    public Batch() {

    }

    public Batch(int batchid, String batchprefix, Timestamp enddate,
List<NumberPlate> plates, boolean availability,
        int price, List<Bid> bids) {
        this.batchid = batchid;
        this.batchprefix = batchprefix;
        this.enddate = enddate;
        this.plates = plates;
        this.availability = availability;
        this.price = price;
        this.bids = bids;
    }
}
```

```
    public Batch(int batchid, String batchprefix, String enddate, List<NumberPlate>
plates, boolean availability,
                int price, List<Bid> bids) {
        this(batchid, batchprefix, Timestamp.valueOf(enddate), plates,
availability, price, bids);
    }

    // Properties

    public int getBatchid() {
        return batchid;
    }

    public void setBatchid(int batchid) {
        this.batchid = batchid;
    }

    public String getBatchprefix() {
        return batchprefix;
    }

    public void setBatchprefix(String batchprefix) {
        this.batchprefix = batchprefix;
    }

    public Timestamp getEnddate() {
        return enddate;
    }

    public void setEnddate(Timestamp enddate) {
        this.enddate = enddate;
    }

    public List<NumberPlate> getPlates() {
        return plates;
    }

    public void setPlates(List<NumberPlate> plates) {
        this.plates = plates;
    }

    public boolean isAvailability() {
        return availability;
    }

    public void setAvailability(boolean availability) {
        this.availability = availability;
    }

    public int getPrice() {
        return price;
    }

    public void setPrice(int price) {
        this.price = price;
    }
}
```

```

    public List<Bid> getBids() {
        return bids;
    }

    public void setBids(List<Bid> bids) {
        this.bids = bids;
    }

    // Methods
}

```

User.java

```

package com.auction.model;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import javax.persistence.*;
import java.util.List;

@Entity
@Table(name = "users")
@JsonIgnoreProperties({"hibernateLazyInitilaizer", "handler"})
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int userid;
    private String companyname;
    private String useremail;
    private String passhash;
    private String firstname;
    private String lastname;
    private String phonenumber;
    private String activationkey;
    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "locationid")
    private Location location;
    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "localofficeid")
    private LocalOffice localoffice;

    @OneToMany(fetch = FetchType.LAZY, targetEntity = Bid.class, mappedBy = "user")
    private List<Bid> bids;

    public User() {

    }

    public User(String companyname, String useremail, String passhash, String
    firstname, String lastname, String phonenumber, String activationkey, Location
    location, LocalOffice localoffice, List<Bid> bids) {
        this.companyname = companyname;
        this.useremail = useremail;
        this.passhash = passhash;
        this.firstname = firstname;
        this.lastname = lastname;
        this.phonenumber = phonenumber;
        this.activationkey = activationkey;
    }
}

```

```
        this.location = location;
        this.localoffice = localoffice;
        this.bids = bids;
    }

    public int getUserId() {
        return userid;
    }

    public void setUserId(int userid) {
        this.userid = userid;
    }

    public String getCompanyname() {
        return companyname;
    }

    public void setCompanyname(String companyname) {
        this.companyname = companyname;
    }

    public String getUseremail() {
        return useremail;
    }

    public void setUseremail(String useremail) {
        this.useremail = useremail;
    }

    public String getPasshash() {
        return passhash;
    }

    public void setPasshash(String passhash) {
        this.passhash = passhash;
    }

    public String getFirstname() {
        return firstname;
    }

    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }

    public String getLastname() {
        return lastname;
    }

    public void setLastname(String lastname) {
        this.lastname = lastname;
    }

    public String getPhonenumber() {
        return phonenumber;
    }

    public void setPhonenumber(String phonenumber) {
        this.phonenumber = phonenumber;
    }
}
```

```

    }

    public String getActivationkey() {
        return activationkey;
    }

    public void setActivationkey(String activationkey) {
        this.activationkey = activationkey;
    }

    public Location getLocation() {
        return location;
    }

    public void setLocation(Location location) {
        this.location = location;
    }

    public LocalOffice getLocaloffice() {
        return localoffice;
    }

    public void setLocalofficeid(LocalOffice localoffice) {
        this.localoffice = localoffice;
    }

    public List<Bid> getBids() {
        return bids;
    }

    public void setBids(List<Bid> bids) {
        this.bids = bids;
    }
}

```

Location.java

```

package com.auction.model;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import javax.persistence.*;
import java.util.List;

@Entity
@Table(name = "locations")
@JsonIgnoreProperties({"hibernateLazyInitilaizer", "handler"})
public class Location {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int locationid;
    private String location;
    private char character;

    @OneToMany(fetch = FetchType.LAZY, targetEntity = User.class, mappedBy =
"location")

```

```

private List<User> users;

public Location() {

}

public Location(int locationid, String location, char character, List<User>
users) {
    this.locationid = locationid;
    this.location = location;
    this.character = character;
    this.users = users;
}

public String getLocation() {
    return location;
}

public void setLocation(String location) {
    this.location = location;
}

public char getCharacter() {
    return character;
}

public void setCharacter(char character) {
    this.character = character;
}

public int getLocationid() {
    return locationid;
}

public void setLocationid(int locationid) {
    this.locationid = locationid;
}

public List<User> getUsers() {
    return users;
}

public void setUsers(List<User> users) {
    this.users = users;
}
}

```

LocalOffice.java

```

package com.auction.model;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import javax.persistence.*;
import java.util.List;

```

```
@Entity
@Table(name = "localoffices")
@JsonIgnoreProperties({"hibernateLazyInitilaizer", "handler"})
public class LocalOffice {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int localofficeid;
    private String localoffice;
    private char[] characters;

    @OneToMany(fetch = FetchType.LAZY, targetEntity = User.class, mappedBy =
"localoffice")
    private List<User> users;

    public LocalOffice() {

    }

    public LocalOffice(int localofficeid, String localoffice, char[] characters,
List<User> users) {
        this.localofficeid = localofficeid;
        this.localoffice = localoffice;
        this.characters = characters;
        this.users = users;
    }

    public String getLocalOffice() {
        return localoffice;
    }

    public void setLocalOffice(String localoffice) {
        this.localoffice = localoffice;
    }

    public char[] getCharacters() {
        return characters;
    }

    public void setCharacters(char[] characters) {
        this.characters = characters;
    }

    public int getLocalofficeid() {
        return localofficeid;
    }

    public void setLocalofficeid(int localofficeid) {
        this.localofficeid = localofficeid;
    }

    public List<User> getUsers() {
        return users;
    }

    public void setUsers(List<User> users) {
```

```

        this.users = users;
    }
}

```

Bid.java

```

package com.auction.model;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import javax.persistence.*;
import java.sql.Timestamp;
import java.util.Objects;

@Entity
@Embeddable
@Table(name = "bids")
@JsonIgnoreProperties({"hibernateLazyInitilaizer", "handler"})
public class Bid implements java.io.Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int bidid;

    @ManyToOne(fetch = FetchType.LAZY, cascade = CascadeType.ALL)
    @JoinColumn(name = "userid")
    private User user;

    @ManyToOne(fetch = FetchType.LAZY, cascade = CascadeType.ALL)
    @JoinColumn(name = "batchid")
    private Batch batch;

    private int price;

    private Timestamp timeofbid;

    public Bid() {

    }

    public Bid( User user, Batch batch, int price, Timestamp timeofbid) {
        this.user = user;
        this.batch = batch;
        this.price = price;
        this.timeofbid = timeofbid;
    }

    public Bid(int bidid, User user, Batch batch, int price, Timestamp timeofbid) {
        this.bidid = bidid;
        this.user = user;
        this.batch = batch;
        this.price = price;
        this.timeofbid = timeofbid;
    }
}

```

```

public int getBidid() {
    return bidid;
}

public void setBidid(int bidid) {
    this.bidid = bidid;
}

public User getUser() {
    return user;
}

public void setUser(User user) {
    this.user = user;
}

public Batch getBatch() {
    return batch;
}

public void setBatch(Batch batch) {
    this.batch = batch;
}

public int getPrice() {
    return price;
}

public Timestamp getTimeofbid() {
    return timeofbid;
}

public void setTimeofbid(Timestamp timeofbid) {
    this.timeofbid = timeofbid;
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;

    if (o == null || getClass() != o.getClass())    return false;

    Bid that = (Bid) o;
    return Objects.equals(batch, that.batch) && Objects.equals(user,
that.user);
}

@Override
public int hashCode() {
    return Objects.hash(batch, user);
}
}

```

ProfanePlate.java

```
package com.auction.model;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import javax.persistence.*;

@Entity
@Table(name="profaneplates")
@JsonIgnoreProperties({"hibernateLazyInitilaizer","handler"})
public class ProfanePlate extends Plate implements java.io.Serializable {

    // Attributes

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int plateid;

    private String numberplate;

    // Constructors

    public ProfanePlate() {

    }

    public ProfanePlate(int plateid, String numberplate) {
        super(plateid, numberplate);
    }

    // Properties

    public int getPlateid() {
        return plateid;
    }

    public void setPlateid(int plateid) {
        this.plateid = plateid;
    }

    public String getNumberplate() {
        return numberplate;
    }

    public void setNumberplate(String numberplate) {
        this.numberplate = numberplate;
    }

    // Methods

}
```

ProfanityChecker.java

```

package com.auction.domain;

import com.auction.model.NumberPlate;
import com.auction.model.ProfanePlate;
import com.auction.repository.ProfanePlateRepository;
import org.springframework.beans.factory.annotation.Autowired;

import java.util.logging.Logger;

public class ProfanityChecker {

    private ProfanePlateRepository profanePlateRepository;

    private static final Logger log =
Logger.getLogger(ProfanityChecker.class.getName());
    @Autowired
    public ProfanityChecker(ProfanePlateRepository profanePlateRepository) {
        this.profanePlateRepository = profanePlateRepository;
    }

    public boolean isProfane(NumberPlate plate) {

        String plateString = plate.getNumberplate();

        Iterable<ProfanePlate> profanePlates = profanePlateRepository.findAll();

        for (ProfanePlate profanePlate : profanePlates) {
            if(plateString.matches(profanePlate.getNumberplate())) {

                log.info("Number plate " + plateString + " is profane.");

                return true;
            }
        }
        return false;
    }
}

```

PriceCalculator.java

```

package com.auction.domain;

import com.auction.model.Batch;
import com.auction.model.NumberPlate;

public class PriceCalculator {

    public PriceCalculator() {

    }

    public int calculatePrice(Batch batch) {

```

```

        int price = 0;
        for (NumberPlate plate : batch.getPlates()) {
            price = price + 15000;
        }
        return price;
    }
}

```

DuplicateChecker.java

```

package com.auction.domain;

import com.auction.model.Batch;
import com.auction.repository.BatchRepository;

import java.util.logging.Logger;

public class DuplicateChecker {

    private BatchRepository batchRepository;

    private static final Logger log =
        Logger.getLogger(BatchPoster.class.getName());

    public DuplicateChecker(BatchRepository batchRepository) {
        this.batchRepository = batchRepository;
    }

    public boolean checkDuplicate(Batch batch) {
        Iterable<Batch> batches = batchRepository.findAll();

        for (Batch currentBatches : batches) {
            if (currentBatches.getBatchprefix().equals(batch.getBatchprefix())) {
                log.info("Duplicate found, check db for duplicate.");
                return true;
            }
        }
        return false;
    }
}

```

BatchAuctionEnd.java

```

package com.auction.domain;

import com.auction.model.Batch;
import com.auction.repository.BatchRepository;
import org.springframework.data.rest.webmvc.RepositoryRestController;
import org.springframework.scheduling.annotation.EnableAsync;
import org.springframework.scheduling.annotation.Scheduled;

import java.sql.Timestamp;

```

```

@EnableAsync
@RepositoryRestController
public class BatchAuctionEnd {
    private BatchRepository batchRepository;

    public BatchAuctionEnd(BatchRepository batchRepository) {
        this.batchRepository = batchRepository;
    }
    /*
    @Scheduled(cron = "0 1 9-17 * * *")

    */
    @Scheduled(cron = "** * * * * *")
    public void makeUnavailable() {
        Iterable<Batch> batches = batchRepository.findAll();
        Timestamp currentTime = new Timestamp(System.currentTimeMillis());
        for (Batch batch : batches) {
            if (currentTime.after(batch.getEnddate())) {
                batch.setAvailability(false);
            }
        }
    }
}

```

BatchGenerator.java

```

package com.auction.domain;

import com.auction.model.Batch;
import com.auction.model.Location;
import com.auction.model.LocalOffice;
import com.auction.model.NumberPlate;

import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Random;

class BatchGenerator {

    private Random rand = new Random();

    public BatchGenerator() {
    }

    private String ageIdentifier(int year, int month) {
        if (month >= 6) year = year + 50;
        return Integer.toString(year).substring(2);
    }

    public Batch batchMaker(Location location, LocalOffice localoffice, int year,
int month, Timestamp enddate) {
        List<Character> chars =
Arrays.asList('A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'R', 'S', 'T',

```

```

        'U','V','W','X','Y','Z');

String stub="";
stub = stub + location.getCharacter();
char[] localchararray = localoffice.getCharacters();
List<Character> localchars = new ArrayList<>();
for (char c : localchararray) {
    if (c!='{' && c!='}' && c!=',' && c!='.') localchars.add(c);
}
stub = stub + localchars.get(rand.nextInt(localchars.size()));
stub = stub + ageIdentifier(year, month);
stub = stub + " ";
stub = stub + chars.get(rand.nextInt(chars.size()));
stub = stub + chars.get(rand.nextInt(chars.size()));

List<NumberPlate> plateList = new ArrayList<>();
Batch batch = new Batch();

batch.setBatchprefix(stub);

for (Character letter : chars) {
    NumberPlate plate = new NumberPlate(stub+letter, batch);
    plateList.add(plate);
}

batch.setPlates(plateList);

enddate.setTime(enddate.getTime());
batch.setEnddate(enddate);
batch.setAvailability(true);
return batch;
}
}

```

BatchPoster.java

```

package com.auction.domain;

import com.auction.model.User;
import com.auction.model.Batch;
import com.auction.model.Bid;
import com.auction.model.NumberPlate;
import com.auction.repository.*;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.rest.webmvc.RepositoryRestController;
import org.springframework.scheduling.annotation.EnableAsync;
import org.springframework.scheduling.annotation.Scheduled;
import org.springframework.transaction.annotation.Transactional;

import java.sql.Timestamp;
import java.time.LocalDate;
import java.util.Iterator;
import java.util.logging.Logger;

@EnableAsync
@RepositoryRestController

```

```

public class BatchPoster {

    private UserRepository userRepository;
    private BatchRepository batchRepository;
    private NumberPlateRepository numberPlateRepository;
    private ProfanePlateRepository profanePlateRepository;
    private BidRepository bidRepository;

    private static final Logger log =
        Logger.getLogger(BatchPoster.class.getName());

    @Autowired
    public BatchPoster(UserRepository userRepository, BatchRepository
batchRepository, BidRepository bidRepository,
        NumberPlateRepository numberPlateRepository,
        ProfanePlateRepository profanePlateRepository) {
        this.userRepository = userRepository;
        this.batchRepository = batchRepository;
        this.numberPlateRepository = numberPlateRepository;
        this.profanityPlateRepository = profanePlateRepository;
        this.bidRepository = bidRepository;
    }

    @Scheduled(cron = "0 0 9-17 * * *")
    @Transactional
    public void postBatches() {

        log.info("begin posting");
        Iterable<User> allUsers = userRepository.findAll();
        BatchGenerator bg = new BatchGenerator();

        log.info("Posting batch to database");
        Timestamp enddate = new Timestamp(System.currentTimeMillis());
        enddate.setTime(enddate.getTime() + 1000 * 60 * 60 * 24 * 14);
        for (User user : allUsers) {
            if (user.getUserid() != 1) {
                Batch newBatch = bg.batchMaker(user.getLocation(),
user.getLocaloffice(), LocalDate.now().getYear(),
                LocalDate.now().getMonthValue(), enddate);

                DuplicateChecker duplicateChecker = new
DuplicateChecker(batchRepository);
                if (!duplicateChecker.checkDuplicate(newBatch)) {

                    ProfanityChecker profanityChecker = new
ProfanityChecker(profanityPlateRepository);

                    for (Iterator<NumberPlate> iter =
newBatch.getPlates().listIterator(); iter.hasNext(); ) {
                        NumberPlate plate = iter.next();
                        if (profanityChecker.isProfane(plate)) {
                            log.info("Number plate " + plate.getNumberplate() + "
has been removed from batch");
                            iter.remove();
                        }
                    }

                    PriceCalculator priceCalculator = new PriceCalculator();

                    newBatch.setPrice(priceCalculator.calculatePrice(newBatch));

```

```

        if (newBatch.getPrice() != 0) {
            log.info("posting batch with prefix " +
newBatch.getBatchprefix());

            batchRepository.save(newBatch);
            log.info("posting plates from batch with prefix " +
newBatch.getBatchprefix());
            numberPlateRepository.save(newBatch.getPlates());

            log.info("Finished posting batch base on user " +
user.getUserid() + " location.");

            log.info("Creating default bid in name of admin");
            User admin = userRepository.getAdmin();
            Bid defaultBid = new Bid(admin, newBatch,
newBatch.getPrice(), new Timestamp(System.currentTimeMillis()));
            bidRepository.save(defaultBid);

        } else {
            log.info("batch with prefix " + newBatch.getBatchprefix() +
"is empty");
        }

        } else {
            log.info("batch with prefix " + newBatch.getBatchprefix() + "
is a duplicate.");
        }
    }
}
}
}
}

```

LocalOfficeRepository.java

```

package com.auction.repository;

import com.auction.model.LocalOffice;
import io.swagger.annotations.Api;
import org.springframework.data.repository.PagingAndSortingRepository;
import org.springframework.data.rest.core.annotation.RepositoryRestResource;

@Api(tags = "Address Entity")
@RepositoryRestResource(collectionResourceRel = "localoffices",
path="localoffices")
public interface LocalOfficeRepository extends
PagingAndSortingRepository<LocalOffice, Integer> {
    LocalOffice findOne(int id);
}

```

LocationRepository.java

```

package com.auction.repository;

import com.auction.model.Location;
import io.swagger.annotations.Api;

```

```

import org.springframework.data.repository.PagingAndSortingRepository;
import
org.springframework.data.rest.core.annotation.RepositoryRestResource;

@Api(tags = "Address Entity")
@RepositoryRestResource(collectionResourceRel = "locations", path="locations")
public interface LocationRepository extends PagingAndSortingRepository<Location,
Integer> {
    Location findOne(int id);

}

```

NumberPlateRepository.java

```

package com.auction.repository;

import com.auction.model.Batch;
import com.auction.model.NumberPlate;
import io.swagger.annotations.Api;
import org.springframework.data.repository.PagingAndSortingRepository;
import org.springframework.data.rest.core.annotation.RepositoryRestResource;

@Api(tags = "Address Entity")
@RepositoryRestResource(collectionResourceRel = "plates", path = "plates")
public interface NumberPlateRepository extends
PagingAndSortingRepository<NumberPlate, Integer> {

    Iterable<NumberPlate> findByBatch(Batch batch);

}

```

ProfanePlateRepository.java

```

package com.auction.repository;

import com.auction.model.ProfanePlate;
import io.swagger.annotations.Api;
import org.springframework.data.repository.PagingAndSortingRepository;
import org.springframework.data.rest.core.annotation.RepositoryRestResource;

import java.util.List;

@Api(tags = "Address Entity")
@RepositoryRestResource(collectionResourceRel = "profaneplates", path =
"profaneplates")
public interface ProfanePlateRepository extends
PagingAndSortingRepository<ProfanePlate, Integer> {

    List<ProfanePlate> findAll();

}

```

BidRepository.java

```

package com.auction.repository;

import com.auction.model.Bid;
import io.swagger.annotations.Api;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.PagingAndSortingRepository;
import org.springframework.data.repository.query.Param;
import org.springframework.data.rest.core.annotation.RepositoryRestResource;
import org.springframework.web.bind.annotation.CorsOrigin;

@Api(tags = "Address Entity")
@CrossOrigin(origins = "http://localhost:63342")
@RepositoryRestResource(collectionResourceRel = "bids", path = "bids")
public interface BidRepository extends PagingAndSortingRepository<Bid, Integer> {

    @Query(value = "select * from bids where price=(SELECT MAX(price) from bids) and batchid=?1", nativeQuery = true)
    Bid GetHighestBidByBatchId(@Param("batchid") int batchid);

}

```

BatchRepository.java

```

package com.auction.repository;

import com.auction.model.Batch;
import com.auction.model.NumberPlate;
import io.swagger.annotations.Api;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.PagingAndSortingRepository;
import org.springframework.data.repository.query.Param;
import org.springframework.data.rest.core.annotation.RepositoryRestResource;
import org.springframework.web.bind.annotation.CorsOrigin;

import java.util.List;

@Api(tags = "Address Entity")
@CrossOrigin(origins = "http://localhost:63342")
@RepositoryRestResource(collectionResourceRel = "batches", path = "batches")
public interface BatchRepository extends PagingAndSortingRepository<Batch, Integer> {
    {
        @Query(value = "select * from batches b where (select count(*) from plates p where p.batchid = b.batchid AND p.numberplate LIKE concat('%',?1,'%'))>0", nativeQuery = true)
        List<Batch> findBatchBySubstring(@Param("substring") String substring);

        @Query(value = "select * from batches b where ABS(price-?1)<=?2 ORDER BY ABS(price-?1) asc", nativeQuery = true)
        List<Batch> findBatchByPrice(@Param("price") int price, @Param("range") int range);

        @Query(value = "select * from batches order by enddate desc LIMIT 4", nativeQuery = true)
        List<Batch> findFourBatchByEnddateDesc();

        @Query(value = "select * from batches order by enddate asc LIMIT 4", nativeQuery = true)

```

```
List<Batch> findFourBatchByEnddateAsc();
```

```
@Query(value = "select * from batches b order by (select count(*) from bids u where  
b.batchid = u.batchid) desc limit 4", nativeQuery = true)  
List<Batch> findFourBatchByInterest();
```

```
}
```

ProfanePlateRepository.java

```
package com.auction.repository;
```

```
import com.auction.model.ProfanePlate;  
import io.swagger.annotations.Api;  
import org.springframework.data.repository.PagingAndSortingRepository;  
import org.springframework.data.rest.core.annotation.RepositoryRestResource;
```

```
import java.util.List;
```

```
@Api(tags = "Address Entity")  
@RepositoryRestResource(collectionResourceRel = "profaneplates", path =  
"profaneplates")  
public interface ProfanePlateRepository extends  
PagingAndSortingRepository<ProfanePlate, Integer> {
```

```
    List<ProfanePlate> findAll();
```

```
}
```

RepositoryConfig.java

```
package com.auction;
```

```
import com.auction.model.*;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.data.rest.core.config.RepositoryRestConfiguration;  
import org.springframework.data.rest.webmvc.config.RepositoryRestConfigurerAdapter;
```

```
@Configuration
```

```
public class RepositoryConfig extends RepositoryRestConfigurerAdapter {
```

```
    @Override  
    public void configureRepositoryRestConfiguration(RepositoryRestConfiguration  
config)  
    {  
        config.exposeIdsFor(NumberPlate.class);  
        config.exposeIdsFor(Batch.class);  
        config.exposeIdsFor(Bid.class);  
        config.exposeIdsFor(LocalOffice.class);  
        config.exposeIdsFor(Location.class);  
        config.exposeIdsFor(ProfanePlate.class);  
        config.exposeIdsFor(User.class);  
    }  
}
```

```
}
```

PlateApplication.java

```
package com.auction;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Import;
import org.springframework.scheduling.annotation.EnableScheduling;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;
import
springfox.documentation.spring.data.rest.configuration.SpringDataRestConfiguration;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

@SpringBootApplication
@EnableSwagger2
@EnableScheduling
@Import({SpringDataRestConfiguration.class})
public class PlateApplication {

    public static void main(String[] args) {
        SpringApplication.run(PlateApplication.class, args);
    }

}
```

Application.Properties

```
spring.datasource.driver-class-name=org.postgresql.Driver
spring.datasource.url=jdbc:postgresql://localhost:5432/platedb
spring.datasource.username=postgres
spring.datasource.password=uwtsd
spring.jpa.database-platform=postgres
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true
spring.jpa.show-sql=false
spring.jpa.open-in-view=false
```

About.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, shrink-
to-fit=no"> <!-- ADD THIS TO EVERY PAGE -->
    <title>About</title>
    <link href="../css/homepage.css" rel="stylesheet" type="text/css"/>

    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
```

```

crossorigin="anonymous">
  <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-q8i/X+965Dz00rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"></script>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9W01clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
crossorigin="anonymous"></script>
  <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy60OrQ6VrjIEaEaff/nJGzIxFDsf4x0xIM+B07jRM"
crossorigin="anonymous"></script>
  <script
src="//ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js"></script>
</head>
<body>
<header>
  
</header>

<ul>
  <li><a href="index.html" target="_self">Home</a></li>
  <li><a href="OngoingAuctions.html" target="_self">Auctions</a></li>
  <li><a class="active" href="About.html" target="_self">About</a></li>
  <li><a href="CreateAccount.html" target="_self">Create Account</a></li>
  <li id="align-right"><a id="islog" target="_self"></a></li>
</ul>

<main role="main">
  <div class="page-subtitle">
    <h1>About this Website</h1>
  </div>
  <div class="textbox">
    This website allows your company to purchase batches of numberplates.
  </div>
  <div class="page-subtitle">
    <h1>What is my location and local office?</h1>
  </div>
  <div class="textbox">
    The location and local office are descriptors used by this website to
    determine your approximate location.
    The location and local office memory tags determine the first and second
    characters of a number plate
    respectively. The possible combinations are described in <a target="_blank"
href="https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attach
ment_data/file/783978/vehicle-registration-numbers-and_number-plates-
inf104.pdf">this document</a>.
    Please note that registrations beginning with the letter K are tagged under
    <i>Milton Keynes</i> by this website.

  </div>

  <div class="page-subtitle">
    <h1>What are batches and how are they created?</h1>
  </div>
  <div class="textbox">
    <p>
      Batches are collections of registration plates, with sizes of up to 24.
      Registration plates with the same
      first 6 characters will be grouped into a batch together with a base
      value of £150. All plates deemed to be
      profane by the DVLA will be removed prior to dispatch to auction.
    </p>

```

```

        </p>
        <p>
            Batches are tailored to the number of users registered. Every hour, a
            batches is created for a large subset
            of site's users. The auction for the batch will end two weeks after
            dispatch to auction.
        </p>
    </div>

</main>
<script src="../../scripts/IsLoggedIn.js"></script>
</body>
<br>
<footer>
</footer>
</html>

```

Batch.html

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, shrink-
to-fit=no"> <!-- ADD THIS TO EVERY PAGE -->
    <title>Batch</title>
    <link href="../../css/homepage.css" rel="stylesheet" type="text/css"/>

    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-q8i/X+965Dz00rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
crossorigin="anonymous"></script>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEeAff/nJGzIxFDsf4x0xIM+B07jRM"
crossorigin="anonymous"></script>
    <script
src="//ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js"></script>

</head>
<body>
<header>
    
</header>

<ul>
    <li><a href="index.html" target="_self">Home</a></li>
    <li><a href="OngoingAuctions.html" target="_self">Auctions</a></li>
    <li><a href="About.html" target="_self">About</a></li>
    <li><a href="CreateAccount.html" target="_self">Create Account</a></li>
    <li id="align-right"><a id="islog" target="_self"></a></li>
</ul>

<main role="main">

```

```

<div class="page-subtitle">
  <h1 id="batchid"></h1>
</div>
<br>
<form class="form">
  <div class="numberplate" id="plate">
  </div>
  <br>
  <div class="info">
    <p>Current Top Bid:
      <span id="bid" style="float:right"></span>
    </p>
  </div>

  <div class="info">
    <p>Auction End Date:
      <span id="enddate" style="float:right"></span>
    </p>
  </div>
</form>
<div class="page-subtitle">
  <h1>Plates in this batch</h1>
</div>
<br>
<div class="flex-container" id="display">
</div>
<br>
<div class="page-subtitle">
  <h1>Place a bid</h1>
</div>
<br>
<form class="form">
  <input id="enterbid" type="number" min="0.00" step="0.01"
placeholder="Enter Bid">
  <br><br>
  <div class="button-wrapper">
    <button type="button" id="bidsubmit">Submit Bid</button>
  </div>
</form>
<script src="../../scripts/BatchPage.js"></script>
<script src="../../scripts/IsLoggedIn.js"></script>
</main>
</body>
<br>
<footer>
</footer>
</html>

```

CreateAccount.html

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0, shrink-
to-fit=no">
  <title>Create Account</title>
  <link href="../../css/homepage.css" rel="stylesheet" type="text/css"/>

  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"

```

```

crossorigin="anonymous">
  <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"></script>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
integrity="sha384-U02eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
crossorigin="anonymous"></script>
  <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaEaff/nJGzIxFDs4x0xIM+B07jRM"
crossorigin="anonymous"></script>
  <script
src="//ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js"></script>
</head>
<body>
<header>
  
</header>

<ul>
  <li><a href="index.html" target="_self">Home</a></li>
  <li><a href="OngoingAuctions.html" target="_self">Auctions</a></li>
  <li><a href="About.html" target="_self">About</a></li>
  <li><a class="active" href="CreateAccount.html" target="_self">Create
Account</a></li>
  <li id="align-right"><a id="islog" target="_self"></a></li>
</ul>

<main role="main">
  <div class="page-subtitle">
    <h1>Create Account</h1>
  </div>
  <br>
  <form id="form">
    <input type="text" placeholder="Enter First Name" id="fname" required>
    <br/><br/>
    <input type="text" placeholder="Enter Last Name" id="lname" required>
    <br/><br/>
    <input type="text" placeholder="Enter Company Name" id="cname"
required>
    <br/><br/>
    <input type="email" placeholder="Enter Email Address" id="email"
required>
    <br/><br/>
    <input type="tel" placeholder="Enter Contact Number"
pattern="\s*\((?020[7,8]{1})\)?[ ]?[1-9]{1}[0-9]{2}[ ]?[0-9]{4})|(0[1-8]{1}[0-
9]{3})?[ ]?[1-9]{1}[0-9]{2}[ ]?[0-9]{3})\s*$" id="number" required>
    <br/><br/>
    <select type="text" name="location" required id="location">
      <option value="" disabled selected>Select your Location</option>
      <option value="1">Anglia</option>
      <option value="2">Birmingham</option>
      <option value="4">Deeside</option>
      <option value="10">Dorset</option>
      <option value="5">Essex</option>
      <option value="7">Fens</option>
      <option value="6">Forest</option>
      <option value="8">Garden of England</option>
      <option value="9">Hampshire</option>
      <option value="12">London</option>
      <option value="13">Manchester</option>
    </select>
  </form>

```

```

        <option value="14">Merseyside</option>
        <option value="11">Milton Keynes</option>
        <option value="15">North</option>
        <option value="16">Oxford</option>
        <option value="17">Preston</option>
        <option value="18">Reading</option>
        <option value="19">Scotland</option>
        <option value="20">Severn Valley</option>
        <option value="3">Wales</option>
        <option value="21">West of England</option>
        <option value="22">Yorkshire</option>
    </select>
    <br/><br/>
    <select type="text" id="localoffice" required>
        <option value="" disabled selected>Select your Local
Office</option>
    </select>
    <br/><br/>
    <input type="password" pattern=".{8,}" placeholder="Enter Password"
id="password" required>
    <br/><br/>
    <input type="password" placeholder="Confirm Password"
id="confirmpassword" required>
    <br/><br/>
    <div class="button-wrapper">
        <button type="submit" id="createaccount">Create Account</button>
    </div>
    <br/>
</form>

</main>

<script src="../../scripts/ValidatePassword.js"></script>
<script src="../../scripts/AppendLocalOffices.js"></script>
<script src="../../scripts/CreateAccount.js"></script>
<script src="../../scripts/IsLoggedIn.js"></script>
</body>
<br>
<footer>
</footer>
</html>

```

Index.html

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, shrink-
to-fit=no"> <!-- ADD THIS TO EVERY PAGE -->
    <title>Homepage</title>
    <link href="../../css/homepage.css" rel="stylesheet" type="text/css"/>

    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"

```

```

integrity="sha384-U02eT0CpHqdSjQ6hJty5KVphtPhzWj9W01clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
crossorigin="anonymous"></script>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy60OrQ6VrjIEaFf/njGzIxFDsf4x0xIM+B07jRM"
crossorigin="anonymous"></script>
    <script
src="//ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js"></script>

</head>
<body>
    <header>
        
    </header>
</div>

    <ul>
        <li><a class="active" href="index.html" target="_self">Home</a></li>
        <li><a href="OngoingAuctions.html" target="_self">Auctions</a></li>
        <li><a href="About.html" target="_self">About</a></li>
        <li><a href="CreateAccount.html" target="_self">Create Account</a></li>
        <li id="align-right"><a id="islog" target="_self"></a></li>
    </ul>

    <main role="main">
        <div class="splashart"></div>
        <div class="page-subtitle">
            <h1>Hottest Batches</h1>
        </div>
        <br>
        <div class="flex-container" id="hottest">

        </div>
        <br><br><br>

        <div class="page-subtitle">
            <h1>Closing Soon</h1>
        </div>
        <br>
        <div class="flex-container" id="closing">
        </div>
        <br><br><br>

        <div class="page-subtitle">
            <h1>Newest Batches</h1>
        </div>
        <br>
        <div class="flex-container" id="newest">
        </div>
    </main>
<script src="../scripts/LoadFrontPage.js"></script>
<script src="../scripts/RedirectBatchPage.js"></script>
<script src="../scripts/IsLoggedIn.js"></script>
</body>
<br>
<footer>
</footer>

```

```
</html>
```

Login.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0, shrink-
to-fit=no"> <!-- ADD THIS TO EVERY PAGE -->
  <title>Log In</title>
  <link href="../css/homepage.css" rel="stylesheet" type="text/css"/>

  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
  <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-q8i/X+965Dz00rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"></script>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
crossorigin="anonymous"></script>
  <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEeAff/nJGzIxFDsf4x0xIM+B07jRM"
crossorigin="anonymous"></script>
  <script
src="//ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js"></script>

</head>
<body>
  <header>
    
  </header>

  <ul>
    <li><a href="index.html" target="_self">Home</a></li>
    <li><a href="OngoingAuctions.html" target="_self">Auctions</a></li>
    <li><a href="About.html" target="_self">About</a></li>
    <li><a href="CreateAccount.html" target="_self">Create Account</a></li>
    <li id="align-right"><a class="active" id="islog" target="_self"></a></li>
  </ul>

  <main role="main">
    <div class="page-subtitle">
      <h1>Log In to your Account</h1>
    </div>
    <br>
    <form id="form">
      <div class="container">
        <input type="email" placeholder="Enter Email" id="email" required/>
        <br/><br/>
        <input type="password" placeholder="Enter Password" id="password"
required/>
      </div>
      <br/>
      <div class="button-wrapper">
        <button type="submit" id="signin">Sign in</button>
```

```

                <br/><br/>
                <a id="forgotpass" type="submit">Forgot your Password?</a>
            </div>
            <br>
        </form>
    </main>
</script>
<script src="../../scripts/Login.js"></script>
    <script src="../../scripts/IsLoggedIn.js"></script>
</body>
<br>
<footer>
</footer>
</html>

```

OngoingAuctions.html

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, shrink-
to-fit=no"> <!-- ADD THIS TO EVERY PAGE -->
    <title>Ongoing Auctions</title>
    <link href="../../css/homepage.css" rel="stylesheet" type="text/css"/>

    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-q8i/X+965Dz00rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
crossorigin="anonymous"></script>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEeAff/nJGzIxFDs4x0xIM+B07jRM"
crossorigin="anonymous"></script>
    <script
src="//ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js"></script>

</head>
<body>
<header>
    
</header>

<ul>
    <li><a href="index.html" target="_self">Home</a></li>
    <li><a class="active" href="OngoingAuctions.html"
target="_self">Auctions</a></li>
    <li><a href="About.html" target="_self">About</a></li>
    <li><a href="CreateAccount.html" target="_self">Create Account</a></li>
    <li id="align-right"><a id="islog" target="_self"></a></li>
</ul>

<main role="main">
    <div class="page-subtitle">
        <h1>Search</h1>

```

```

    </div>
    <br>
    <form id="form">
        <label>Find by Number Plate</label>
        <input type="text" name="Number Plate" id="substring" placeholder="Enter
Number Plate">
        <br><br>
        <div class="button-wrapper">
            <button id="searchbysubstring" type="submit">Search</button>
        </div>
        <br>
        <div class="blackline"></div>
        <br>
        <label>Find by Price</label>
        <input type="number" min="0.00" step="0.01" placeholder="Enter price"
id="price">
        <br><br>
        <input type="number" min="0.00" step="0.01" placeholder="Enter range"
id="range">
        <br><br>
        <div class="button-wrapper">
            <button id="searchbyprice" type="submit">Search</button>
        </div>
        <br>
    </form>
    <div class="flex-container" id="batch-list">

</div>
<div id="pages">
    <p>Page Number</p>
</div>
<script src="../../scripts/Search.js"></script>
<script src="../../scripts/GetBatches.js"></script>
<script src="../../scripts/AppendLocalOffices.js"></script>
<script src="../../scripts/RedirectBatchPage.js"></script>
<script src="../../scripts/IsLoggedIn.js"></script>
</main>
</body>
<br>
<footer>
</footer>
</html>

```

AppendLocalOffice.js

```

$(document).ready(function() {
    $("#location").change(function() {

        var el = $(this);

        var localoffices;
        if (el.val() === "1" ) {

            localoffices= '<option value="3">Ipswich</option>' +
                '<option value="2">Norwich</option>' +
                '<option value="1">Peterborough</option>';

```

```

} else if (el.val()=== "2") {
    localoffices= '<option value="4">Birmingham</option>';

} else if (el.val()=== "3") {
    localoffices =
        '<option value="9">Bangor</option>'
        + '<option value="5">Cardiff</option>' +
        '<option value="8">Swansea</option>' ;

} else if (el.val()=== "4") {
    localoffices = '<option value="10">Chester</option>' +
        '<option value="11">Shrewsbury</option>';

} else if (el.val()=== "5") {
    localoffices = '<option value="12">Chelmsford</option>';

} else if (el.val()=== "6") {
    localoffices = '<option value="14">Lincoln</option>'+
        '<option value="13">Nottingham</option>'
        ;
} else if (el.val()=== "7") {
    localoffices = '<option value="14">Lincoln</option>'+
        '<option value="13">Nottingham</option>'
        ;

} else if (el.val()=== "8") {
    localoffices = '<option value="16">Brighton</option>'+
        '<option value="15">Maidstone</option>';

} else if (el.val()=== "9") {
    localoffices = '<option value="17">Bournemouth</option>' +
        '<option value="18">Portsmouth</option>';
} else if (el.val()=== "10") {
    localoffices = '<option value="17">Bournemouth</option>' +
        '<option value="18">Portsmouth</option>';

} else if (el.val()=== "11") {
    localoffices = '<option value="19">Borehamwood</option>' +
        '<option value="21">Northampton</option>';

} else if (el.val()=== "12") {
    localoffices =
        '<option value="23">Borehamwood</option>' +
        '<option value="25">Sidcup</option>' +
        '<option value="22">Wimbledon</option>';

```

```

    } else if (el.val() === "13") {
        localoffices = '<option value="26">Manchester</option>';
    } else if (el.val() === "14") {
        localoffices = '<option value="26">Manchester</option>';

    } else if (el.val() === "15") {
        localoffices = '<option value="6">Newcastle</option>' +
            '<option value="27">Stockton</option>';

    } else if (el.val() === "16") {
        localoffices = '<option value="7">Oxford</option>';

    } else if (el.val() === "17") {
        localoffices = '<option value="29">Carlisle</option>' +
            '<option value="28">Preston</option>';

    } else if (el.val() === "18") {
        localoffices = '<option value="30">Reading</option>';

    } else if (el.val() === "19") {
        localoffices = '<option value="34">Aberdeen</option>' +
            '<option value="33">Dundee</option>' +
            '<option value="32">Edinburgh</option>' +
            '<option value="31">Glasgow</option>' +
            '<option value="35">Inverness</option>';

    } else if (el.val() === "20") {
        localoffices = '<option value="36">Worcester</option>';

    } else if (el.val() === "21") {
        localoffices = '<option value="39">Bristol</option>' +
            '<option value="37">Exeter</option>' +
            '<option value="38">Truro</option>';

    } else if (el.val() === "2") {
        localoffices =
            '<option value="42">Beverley</option>' +
            '<option value="40">Leeds</option>' +
            '<option value="41">Sheffield</option>';

    }
    $("#localoffice").empty().append(localoffices);

});
});

```

BatchPage.js

```

$(document).ready(function() {

    var url = String(window.location.href);
    var split = url.split("=");
    var batchid = split[1];

    var batch;

```

```

$.ajax({
  url: "http://localhost:8080/batches/" + batchid,
  type: 'GET',
  contentType: 'application/json; charset=utf-8',
  success: function (resultData) {

    batch = resultData;
    console.log(batch);

    const title = document.getElementById("batchid");
    title.textContent = "Batch #" + resultData.batchid;

    const plate = document.getElementById("plate");
    plate.textContent = resultData.batchprefix + "\u2981";

    const bid = document.getElementById("bid");
    var pricestring = String(resultData.price);
    var num = pricestring.substring(0, pricestring.length-2);
    var dec = pricestring.substring(pricestring.length-2,
    pricestring.length);

    bid.textContent = "£" + num + "." + dec;

    const end = document.getElementById("enddate");
    end.textContent = resultData.enddate.substring(11, 19) + " " +
    resultData.enddate.substring(0,10);

    $.ajax({
      url: "http://localhost:8080/batches/" + batchid + "/plates",
      type: 'GET',
      contentType: 'application/json; charset=utf-8',
      success: function (resultData) {

        const display = document.getElementById("display");

        var plates = resultData._embedded.plates;
        plates.forEach(plate => {
          const platedisplay = document.createElement('div');
          platedisplay.setAttribute('class', 'numberplate');
          platedisplay.textContent = plate.numberplate;
          display.appendChild(platedisplay);
        })
      }
    })
  },
  error: function(jqXHR, textStatus, errorThrown) {
    console.log("LoadBatchPage");
    console.log("jqXHR: " + jqXHR +
      "\r\n status: " + textStatus
      + "\r\n error: " + errorThrown
      + "\r\n requested url: " + "http://localhost:8080/batches/" +
batchid);
  },
});

$('#bidsubmit').click(function(e) {
  e.preventDefault();

```

```

        if(sessionStorage.getItem('1')===null) {
            window.alert("Please log in or create an account before bidding");
        }
        var bidvalue = document.getElementById("enterbid").value*100;
        var title = document.getElementById("batchid").textContent;
        var batchidd = title.split('#')[1];
        console.log(batchidd);
        $.ajax({
            url:
'http://localhost:8080/bids/search/GetHighestBidByBatchId?batchid=' + batchidd,
            type: 'GET',
            crossDomain: true,
            contentType: 'application/json; charset=utf-8',
            success: function (resultData) {
                const price = resultData.price;

                if (price >= bidvalue) {
                    window.alert("Please enter a value greater than " +
price/100);
                } else {
                    var timestamp = new Date();
                    var bid = {batch
: 'http://localhost:8080/batches/'+batchidd, user : 'http://localhost:8080/users/'+
String(sessionStorage.getItem('1')), price : bidvalue, timeofbid:
timestamp.toISOString()};

                    console.log("BID BEING SENT" +JSON.stringify(bid));
                    $.ajax({
                        url: "http://localhost:8080/bids",
                        type: 'POST',
                        contentType: 'application/json',
                        crossDomain: true,
                        data: JSON.stringify(bid),
                        success: console.log("posted bid"),
                        error: function(err) {
                            console.log("Error posting bid: " + String(err));
                        }
                    });

                    batch.price = bidvalue;
                    $.ajax({
                        url: "http://localhost:8080/batches/"+batchid,
                        type: 'PUT',
                        contentType: 'application/json',
                        crossDomain: true,
                        data: JSON.stringify(batch),
                        success: console.log("updated batch"),
                        error: function(err) {
                            console.log("Error updating batch: " +
String(err));
                        }
                    });
                }
            },
            error: function(err) {
                console.log("Error getting highest bid");
            }
        })
    });
});

```

CreateAccount.js

```
$(document).ready(function() {

    $.ajax({
        url: 'http://localhost:8080/users',
        type: 'GET',
        crossDomain: true,
        contentType: 'application/json; charset=utf-8',
        success: function(resultData) {
            var users = resultData._embedded.users;

            $('#createaccount').click(function(e) {
                e.preventDefault();

                const companyname = document.getElementById('cname');
                const firstname = document.getElementById('fname');
                const lastname = document.getElementById('lname');
                const email = document.getElementById('email');
                const number = document.getElementById('number');
                const password = document.getElementById('password');
                const location = document.getElementById('location');
                const localoffice = document.getElementById('localoffice');

                var data = { companyname : companyname.value,
                    firstname :  firstname.value ,
                    lastname : lastname.value,
                    useremail :  email.value ,
                    passhash :  password.value,
                    activationkey : 'reset123',
                    phonenumber :  number.value,
                    location : 'http://localhost:8080/locations/' + location.value,
                    localoffice : 'http://localhost:8080/localoffices/' +
localoffice.value };

                for (var i=0; i<users.length; i++) {
                    if (users[i].useremail === email.value) {
                        console.log()
                        window.alert("This email is already registered");
                    }
                }
                $.ajax({
                    type: 'POST',
                    contentType: 'application/json',
                    url: 'http://localhost:8080/users',
                    crossDomain: true,
                    data: JSON.stringify(data),
                    dataType: 'json',
                    timeout: 600000,
                    success: console.log("posted" + JSON.stringify(data)),
                    error: function (err) {
                        if (err.readyState === 0) {
                            window.alert("This email is already registered");
                        }
                    }
                });
                $('#form')[0].reset();
            });
        },
        error: function(err) {
```

```

        window.alert("cannot register users at this time");
    /*
        window.location.href = "index.html"; */
    }
    });

})

```

GetBatches.js

```

function getFromUrl(url) {
    $.ajax({
        url: url,
        type: 'GET',

        contentType: 'application/json; charset=utf-8',
        success: function(resultData) {
    /*

    */

        var batches = resultData._embedded.batches;

        const list = document.getElementById('batch-list');

        while(list.firstChild) {
            list.removeChild(list.firstChild);
        }

        batches.forEach(batch => {
            const item = document.createElement('div');
            item.setAttribute('class', 'flex-item');

            const plate = document.createElement('div');
            plate.setAttribute('class', 'numberplate');
            plate.textContent = batch.batchprefix + "\u2981";

            const price = document.createElement('p');
            price.textContent = "Top Bid:";

            const pricevalue = document.createElement('span');
            pricevalue.setAttribute('style', 'float:right');
            const pricestring = batch.price.toString();
            var num = pricestring.substring(0, pricestring.length-2);
            var dec = pricestring.substring(pricestring.length-2,
pricestring.length);
            pricevalue.textContent = "£" + num + "." + dec;
            price.appendChild(pricevalue);

            const ending = document.createElement('p');
            var date = batch.enddate;
            ending.textContent = "Ending:";

            const endingdate = document.createElement('span');
            endingdate.setAttribute('style', 'float:right');
            endingdate.textContent = date.substring(11, 19) + " " +
date.substring(0,10);
            ending.appendChild(endingdate);

```

```

        const butt = document.createElement('button');
        butt.setAttribute('class', 'seemore');
        butt.setAttribute('type', 'submit');
        butt.setAttribute('id', batch.batchid);
        butt.textContent = "See more";

        item.appendChild(plate);
        list.appendChild(item);
        item.appendChild(price);
        item.appendChild(ending);
        item.appendChild(butt);
    });
    if(list.firstChild === null) {
        const empty = document.createElement('p');
        empty.setAttribute('style', 'color:#ffffff');
        empty.textContent = "No batches found";
        list.appendChild(empty);
    }
},
error : function(jqXHR, textStatus, errorThrown) {
    console.log("GetBatches");
    console.log("jqXHR: " + jqXHR +
        "\r\n status: " + textStatus
        + "\r\n error: " + errorThrown);
},
timeout: 120000,
});
}

```

IsLoggedIn.js

```

$(document).ready( function() {
    const log = document.getElementById("islog");

    if(sessionStorage.getItem('l') !== null) {
        log.textContent = "Log Out";
        log.setAttribute('href', 'Login.html');
        log.setAttribute('target', '_self');
    } else {
        log.setAttribute('href', 'Login.html');
        log.setAttribute('target', '_self');
        log.textContent = "Log In";
    }

    $("#islog").click(function(e) {
        if(sessionStorage.getItem('l') !== null) {
            e.preventDefault();
            sessionStorage.clear();
            window.location.replace("index.html");
        }
    })
});

```

LoadFrontPage.js

```

$(document).ready(function () {

load4("http://localhost:8080/batches/search/findFourBatchByInterest","hottest");

load4("http://localhost:8080/batches/search/findFourBatchByEnddateDesc","newest");

load4("http://localhost:8080/batches/search/findFourBatchByEnddateAsc","closing");

});

function load4(url, section) {
    $.ajax({
        url: url,
        type: 'GET',

        contentType: 'application/json; charset=utf-8',
        success: function (resultData) {

            const tofill = document.getElementById(section);
            var batches = resultData._embedded.batches;

            batches.forEach(batch => {
                const item = document.createElement('div');
                item.setAttribute('class', 'flex-item');

                const plate = document.createElement('div');
                plate.setAttribute('class', 'numberplate');
                plate.textContent = batch.batchprefix + "\u2981";

                const price = document.createElement('p');
                price.textContent = "Top Bid:";

                const pricevalue = document.createElement('span');
                pricevalue.setAttribute('style', 'float:right');
                const pricestring = batch.price.toString();
                var num = pricestring.substring(0, pricestring.length - 2);
                var dec = pricestring.substring(pricestring.length - 2,
pricestring.length);
                pricevalue.textContent = "£" + num + "." + dec;
                price.appendChild(pricevalue);

                const ending = document.createElement('p');
                var date = batch.enddate;
                ending.textContent = "Ending:";

                const endingdate = document.createElement('span');
                endingdate.setAttribute('style', 'float:right');
                endingdate.textContent = date.substring(11, 19) + " " +
date.substring(0, 10);
                ending.appendChild(endingdate);

                const butt = document.createElement('button');
                butt.setAttribute('class', 'seemore');
                butt.setAttribute('type', 'submit');
                butt.setAttribute('id', batch.batchid);

                butt.textContent = "See more";
            });
        }
    });
}

```

```

        item.appendChild(plate);
        tofill.appendChild(item);
        item.appendChild(price);
        item.appendChild(ending);
        item.appendChild(butt);
    });

    },
    error: function (jqXHR, textStatus, errorThrown) {
        console.log("LoadFrontPage");
        console.log("jqXHR: " + jqXHR +
            "\r\n status: " + textStatus
            + "\r\n error: " + errorThrown);
    }
});
}

```

Login.js

```

$('#signin').click(function(e) {
    e.preventDefault();

    var email = document.getElementById("email").value;
    var pass = document.getElementById("password").value;

    $.ajax({
        url: 'http://localhost:8080/users',
        type: 'GET',
        crossDomain: true,
        contentType: 'application/json; charset=utf-8',
        success: function (resultData) {
            var users = resultData._embedded.users;
            var success = false;
            for (var i=0; i<users.length; i++) {
                if (String(users[i].useremail) === email) {
                    if (users[i].passhash === pass) {
                        // Login
                        success = true;
                        sessionStorage.setItem('1',users[i].userid);
                        sessionStorage.setItem('2',email)
                        console.log("Success!" +
sessionStorage.getItem(users[i].userid));
                        window.location.replace("index.html");
                    }
                }
            }
            if (success === false) {
                window.alert("login not successful");
            }
        }
    });
});

```

RedirectBatchPage.js

```

$(document).on('click', ".seemore", function(e) {
    e.preventDefault();
    var batchid = this.id;

```

```

$.ajax({
    url: "http://localhost:8080/batches/" + batchid,
    type: 'GET',

    contentType: 'application/json; charset=utf-8',
    success: function (resultData) {
        console.log(resultData);
        window.location.replace('batch.html?batchid=' + batchid);
    },
    error: function(jqXHR, textStatus, errorThrown) {
        console.log("RedirectBatchPage");
        console.log("jqXHR: "+jqXHR+
            "\r\n status: "+textStatus
            + "\r\n error: " + errorThrown
            +"\r\n requested url: " + "http://localhost:8080/batches/" +
batchid);
    },
});
})

```

Search.js

```

$(document).ready(function() {
    if (document.getElementById("substring").value==="" &&
document.getElementById("price").value=== "") {
        getFromUrl("http://localhost:8080/batches");
    }
});

$("#searchbysubstring").click(function(e) {
    e.preventDefault();
    var url =
"http://localhost:8080/batches/search/findBatchBySubstring?substring=";

    var substring = document.getElementById("substring").value.toUpperCase();

    substring = substring.replace(/ /g, "%20");
    url = url + substring;

    getFromUrl(url);
});

$("#searchbyprice").click(function(e) {
    e.preventDefault();
    var url = "http://localhost:8080/batches/search/findBatchByPrice?";
    var price = document.getElementById("price").value;
    var range = document.getElementById("range").value;

    if(range=== "") {
        url = url + "price=" + price * 100 + "&range=" +10000000;
    } else {
        url = url + "price=" + price * 100 + "&range=" + range * 100;
    }
    getFromUrl(url);
});

```

ValidatePassword.js

```
var password = document.getElementById("password")
    , confirm_password = document.getElementById("confirmpassword");

function validatePassword(){
    if(password.value != confirm_password.value) {
        confirm_password.setCustomValidity("Passwords Don't Match");
    } else {
        confirm_password.setCustomValidity('');
    }
}

password.onchange = validatePassword;
confirm_password.onkeyup = validatePassword;
```

Homepage.css

```
body {
    font-family : "Arial";
    margin: 0px;
    background-color: #333333;
}
header {
    background-color: #ffffff;
}

.flex-container {
    display: flex;
    height: 25%;
    flex-wrap: wrap;
    align-items: center;
    justify-content: center;
}

.flex-item {
    width: 25rem;
    height: 290px;
    margin: 10px;

    padding: 3rem;
    background-color: #666666;
}

.splashart {
    background-image: url(../images/splashart.jpg);
    background-position: center;
    background-repeat: no-repeat;
    background-size: 100% 100%;
    filter: blur(7px);
    width: 100%;
    -webkit-filter: blur(7px);
    min-height: 720px;
}

main {
    background-color: #333333;
```

```
padding: 3rem;
align-items: center;
}

.page-subtitle {
color: #ffffff;
border-bottom: 1px solid #eab507;
}

footer {
background-color: #eab507;
bottom: 0;
width: 100%;
height: 2.5rem;
}

ul {
list-style-type: none;
overflow-x: hidden;
background-color: #eab507;
padding: 0px 0px;
width: 100%;
white-space: nowrap;
text-decoration: none;
}

#align-right {
float: right;
}

li {
display: inline;
float: left;
}

li a {
display: block;
color: white;
text-align: left;
padding: 15px 17px;
text-decoration: none;
}

li a:hover {
background-color: #ddd;
text-decoration: none;
color: #000000;
}

.active {
background-color: #000000;
}

@font-face {
font-family: "Plate";
src: url("../platefont/uknumberplate-webfont.eot");
src: url("../platefont/uknumberplate-webfont.ttf") format("truetype");
src: url("../platefont/uknumberplate-webfont.woff") format("woff");
}
```

```
    src: url("../platefont/uknumberplate-webfont.woff2") format("woff2");
}

.numberplate {
    border: 2px solid black;
    font-family: "Plate" ;
    font-size: 3em;
    text-align: center;
    color: #000000;
    max-width: 15rem;
    margin: auto;
    background-color: #eab507;
}

form {
    height: auto;
    background-color: #eab507;
    margin: auto;
    color: #ffffff;
    padding: 2rem;
    max-width: 30rem;
}

input {
    width: 100%;
    border: 3px solid transparent;
    height: 2.5rem;
}

input:focus {
    border : 3px solid #000000;
    transition: 0.5s;
}

select, option {
    width: 100%;
    height: 2.5rem;
}

button {
    display: inline-block;
    border: 2px solid #000000;
    padding: 1rem 2rem;
    margin: 0;
    text-decoration: none;
    background: #b88808;
    color: #ffffff;
    cursor: pointer;
    text-align: center;
    transition: background 250ms ease-in-out,
    transform 150ms ease;
}

button:hover{
    color:#000000;
    background-color: #ffffff;
}

.button-wrapper {
    text-align: center;
```

```
}

#forgotpass {
  text-decoration: underline;
}

.textbox{
  background-color: #eab507;
  margin: 5rem;
}

#pages{
  color: #ffffff;
}

.blackline {
  min-height: 1px;
  border-bottom: 1px solid #000000;
}

input {
  width: 100%;
}

input[type='number'] {
  -moz-appearance: textfield;
}

input[type=number]::-webkit-inner-spin-button,
input[type=number]::-webkit-outer-spin-button {
  -webkit-appearance: none;
  margin: 0;
}
```