Cached services

Contents

[1. Architecture 1](#_Toc480789473)

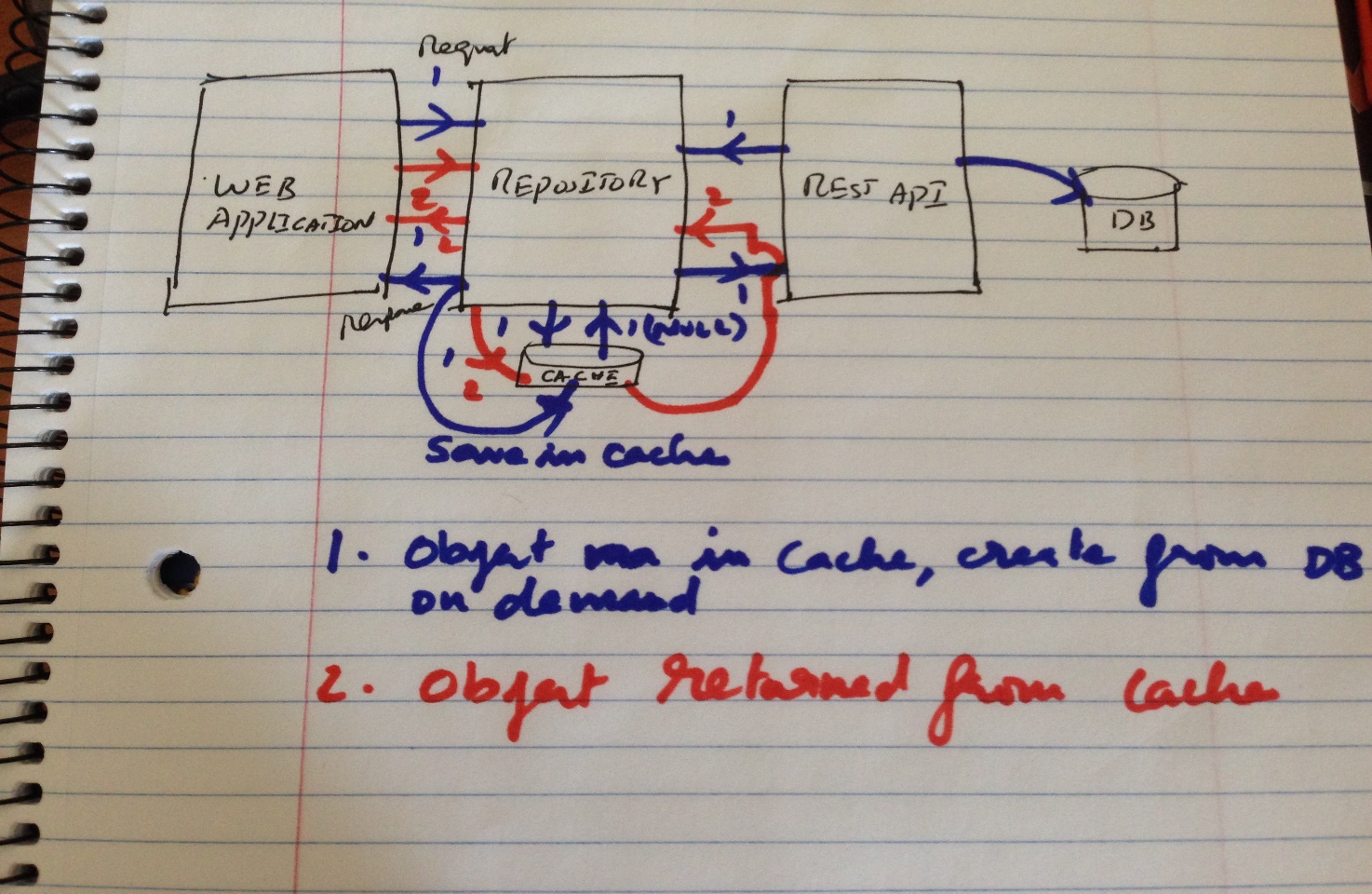
[a) Database 1](#_Toc480789474)

[b) REST API 1](#_Toc480789475)

[c) Repository with client 1](#_Toc480789476)

[2.Running the setup 2](#_Toc480789477)

# Architecture



## Database

Database name: Hackerrank, Table: Job.

Created in Sql Server

## REST API

App Name:Hackerrank\_WebAPI.

MVC app, Entity framework, Data first approach

## Repository with client

App name: HR\_Repository\_Cached\_Services

MVC , HTTP Context cache, multithreading with Async task, Razor

# 2. Running the setup

1. Create a DB Hackerrank
2. Run the DBscripts to create the Job table & insert the values or restore the backed up DB.
3. Update JobModel.edmx to the new DB environment-> test connection-> update web.config connection string.
4. The JobModel.cs is rewritten add in the file

public class JobModel

{

public int Job\_id { get; set; }

public string CustomerName { get; set; }

public string Location { get; set; }

public string TechniciansAssigned { get; set; }

public decimal Estimate { get; set; }

}

1. Start the Hackerrank\_WebAPI
2. Test on browser with <http://localhost:portno/api/job>, returns the XML response
3. Change the string uri = <http://localhost:xxxx/api/job> in RepositoryCache.cs in HR\_Repository\_Cached\_Services App.
4. Start the HR\_Repository\_Cached\_Services app(client).
   1. Initially the data is not available on Cache, the repository creates a new object calling the API, stores it in the cache & returns the result to the client
   2. The sliding timeout is set to 5 minutes, if the client requests for the object within the 5 minutes then the object is returned from Http Context cache.
   3. If the client calls after 5 minutes then a new object is created calling the API, stored in the cache & returned to the client
   4. If the API is stopped & the cache has time out then an error page is returned.
5. The context Key, Value is removed if any CRUD operations are performed in DB to sync cache with DB.