



JOBHUB

DREAM API & Framework EBook

PRODUCT SHOWCASE

This EBook utilizes the DREAMAPI and framework for building a Job search portal hub similar to GlassDoor. We will use a custom application development framework to build JOBHUB Suite of Apps.

Jason Azim

Email: me@www.improvecode.com

Visit: www.improvecode.com

JobHub - DREAM API & Framework EBook

Table of Contents

1 – Introduction to JobHub Case Study :	2
2 – DREAM API and Framework :	3
3 – JobHub Database Design :	4
3a – Setup JobHub MiddleTier Data Access Layer :	5
4 – Building the DREAM API :	5
4a – Define Data Entity Classes :	5
4b – Shape Classes :	6
4c – Define Repository Classes :	6
5 – JobHub StoreFront :	7
6 – JobHub Desktop Admin App :	12
6 – JobHub Cloud Service :	16
Appendix :	18

JobHub - DREAM API & Framework EBook

1 – Introduction to JobHub Case Study :

In this EBook we will be building a Job search management and recruitment online tool similar to GlassDoor called JobHub. JobHub is an online portal where recruiters can connect with potential job seekers and job descriptions are composed online and can then be emailed out to candidates who subscribe to the portal. *JobHub is a suite of applications for matching the job skills of candidates with the job skills that are mentioned on the job description. This matching is done with machine learning and automation.*

We will be building many different versions of the JobHub portal :

1. **JobHub Store Front:** The first version will be built using AMD and require js. We will create components using javascript DREAM API custom Framework. You can read more about the DREAM API and Framework later on in this document when we start building the application step by step. In this step we will build the internet portal as a Store Front with microsities inside it and each site can be constructed as a separate application / service. This is for **external** customers and subscribers.
2. **JobHub Desktop Admin App:** After building the first version of the JobHub we will then migrate it to the latest angular version with Visual Studio 2019 and we will build an admin desktop application to manage the Store Front. This time around we will utilize the material design specification and we will utilize templates and build the portal more as an Application and this app will be used by our **internal** employees. So we will remove the Store Front and marketing nature of the JobHub portal that we implemented in Step 1 above. This version of JobHub portal will be built using .NET Standard Framework and .NET Core so that it can run on Ubuntu or Windows 10.
3. **JobHub Cloud Service:** Next we will recreate the angular version of JobHub portal using REACT to show how we can migrate the DREAM Framework from one environment to another and this time we will built the application as a cross platform asp.net core application which can be run in the cloud and this version of the application will be used by our partners and integration partners and vendors etc.

Note: All three versions of the code base are available on Gitlab.

And like this we can build the JOBHUB App using many different types of technology stacks. Look at the appendix section for more details on choosing a framework and building an app using swappable layers of technology.

JobHub - DREAM API & Framework EBook

2 – DREAM API and Framework :

The **DREAM** API is a custom development methodology. It is an abbreviation for:

D – Data Entity Class – A class that interfaces with the data layer and manages data interactions.

R – Render Shape Class – A user interface layer component that accepts user interaction.

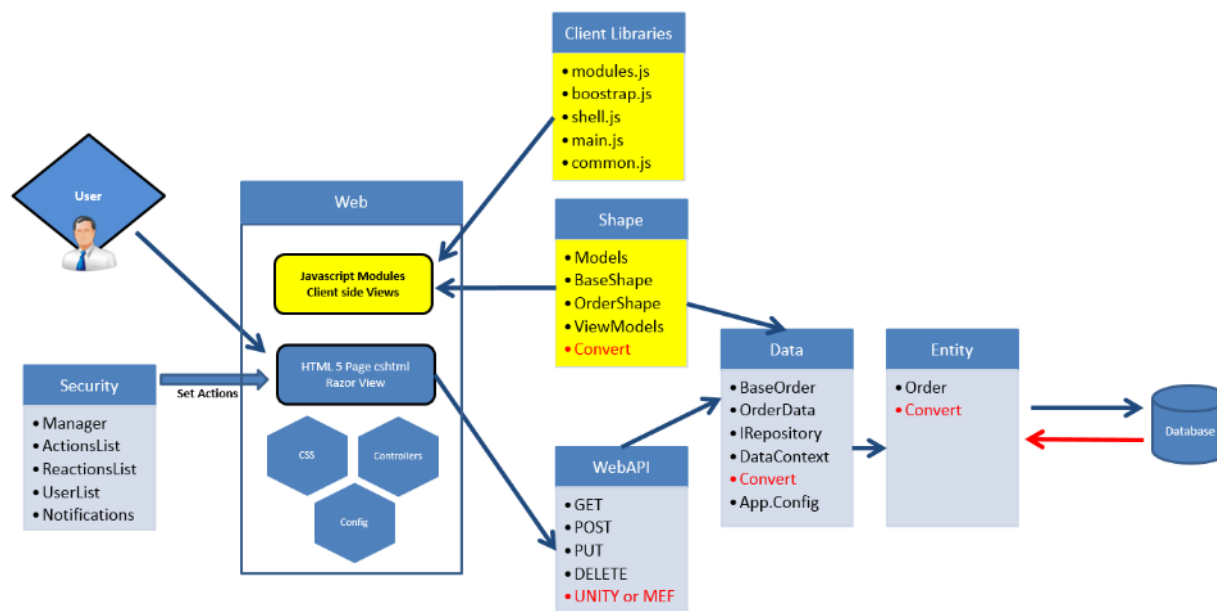
E – Enterprise Asset Management. A connective framework for managing assets of a company.

A – A backend API that is used to serve the data as a service.

M – Building the software components as modules / sub packages / libraries.

NOTE: A description of EAM and enterprise asset management and enterprise asset integration is beyond the scope of this book and we can provide more information regarding this in a White Paper written to address the workflows, orchestrations and integration concepts that make up this framework.

So basically any application that supports the DREAM API will utilize the layers mentioned above. Here is an architecture diagram showing the DREAM Framework in action.

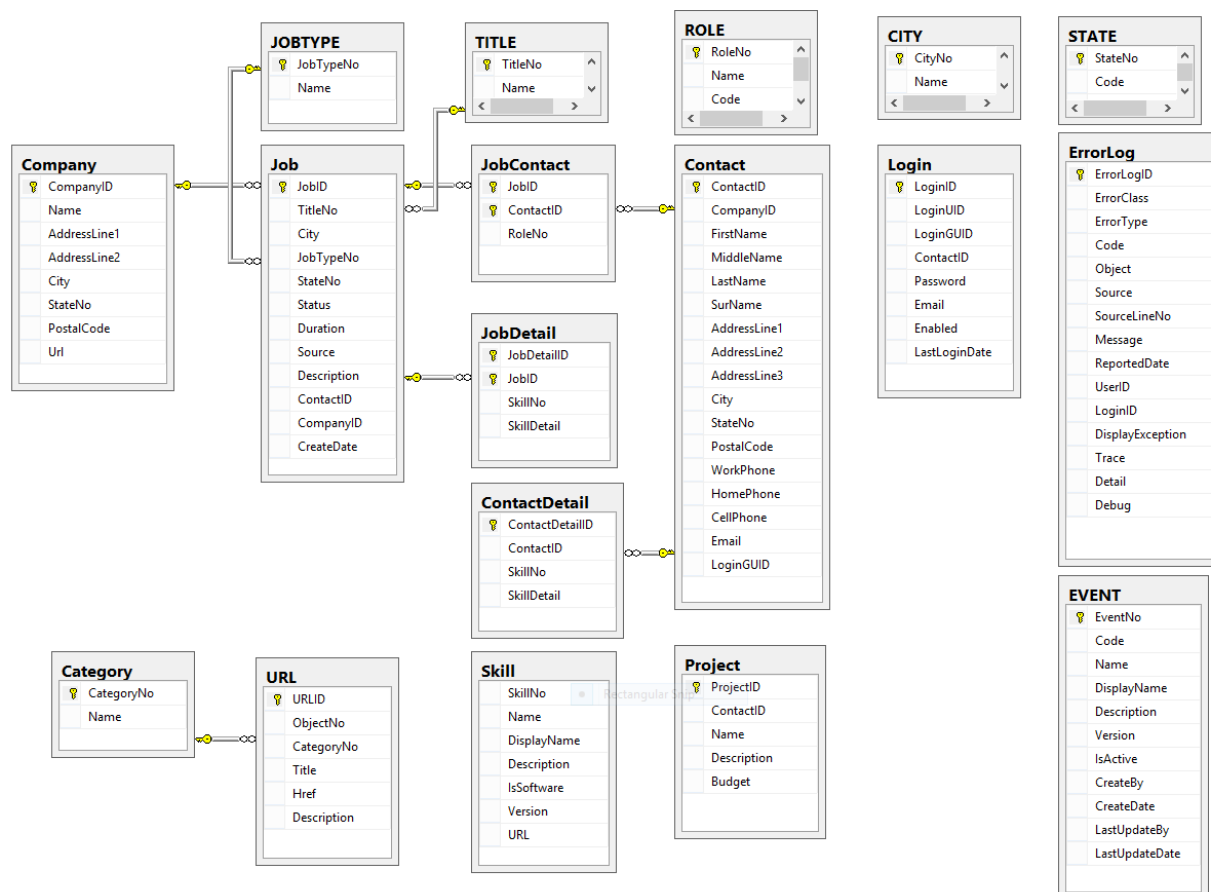


Next we will be applying the above architecture diagram to build the three different versions of the JobHub Recruitment portal which is similar to GlassDoor. Before we can start building the different versions of the JobHub portal we need to go over a basic database model that we are trying to implement.

JobHub - DREAM API & Framework EBook

3 – JobHub Database Design :

The JobHub database diagram is shown below with database relationships and tables.



A Company has many Jobs. A Job has many different types of Contacts that are trying to fulfill the Job Opening with candidates so a role type could be recruiter or candidate. The Job has a job description and skill set that recruiters are looking for in candidates. The skills for the candidates need to be matched with the skills for the job description for a successful placement. A job contact can have projects with a budget also that they might do a fix bid on for contract work. We also need to be able to track the location and addresses for contacts and companies and be able to send out email notifications. The JobHub system matches candidate skills with the job description skills as a one to one mapping and generates a scorecard for how suitable the match is thus making the job search quicker and more automatic. The system will generate a KPI of which candidates match with jobs and this meter can be used by both the job seeker and job provider to quickly fill the role. The system can be used to build an online resume also which is searchable using skills. The upload and download and live editing of the online resume is not supported at the moment.

NOTE: The database is available on MySQL and SQL Server. The middle tier data access mechanism we will be using can access data from both databases. To enhance security the StoreFront is stored on MySQL and the admin data is stored on SQL Server.

JobHub - DREAM API & Framework EBook

3a – Setup JobHub MiddleTier Data Access Layer :

As mentioned in the previous section the first thing we need to do is setup a middle tier database access mechanism. *We want to be able to store the data on MySql or SqlServer* so we will be utilizing the System.Data namespace of the .NET Framework to allow us to retrieve data. We will not be using Entity Framework because our data layer needs to be portable so we can run it on Linux or Windows and we will tie it to our session state and security using redis.



The JSON representations of the database data will be stored in redis for caching and quick retrieval and to speed up things. There is a redis cloud service available to provide this interface or we may set one up locally and use it along with Azure. MySQL can be installed for free and we were using version 6.3 or the MySQL workbench at the time of writing of this book along with MS SQL Server 2016.

4 – Building the DREAM API :

In this section we will apply the concepts of DREAM Framework step by step. These steps do not have to be performed in the sequence shown below. If some requirements or information is missing then we can fill in the information that we know and skip the sections that we don't know to keep moving forward.

4a – Define Data Entity Classes :

The Data Entity classes are of two types Core and Lookup. The core classes contain the actual data. Each data entity class maps to a corresponding database table with the same name and has the same number fields as in the database table. The application will use the following data entity classes:

1. CityEntity – Lookup class for storing City data.
2. ContactEntity – Core class for storing Contact data.
3. ErrorLogEntity – This class is returned in case of an api error along with http error codes.
4. LoginEntity – Core class used for user security authentication and authorization.
5. ProjectEntity – Core class for storing project data.
6. URLEntity – Core class for storing urls.
7. JobEntity – Core class for storing Job data.
8. JobTypeEntity – Lookup class for storing job types.
9. RoleEntity – Lookup class for storing role types.
10. StateEntity – Lookup class for storing state codes.
11. TitleEntity – Lookup class for storing titles.

JobHub - DREAM API & Framework EBook

4b – Shape Classes :

Each data entity class will have a corresponding shape class. The word shape means user interface (UI) component layer class. For example the CityEntity class will have a corresponding CityShape class. The reason for having this additional layer is because we want to augment the fields that are brought in from the database with our own calculated fields, columns and properties. For example the zip code for city might be stored as a number but in the shape class it might be stored as a two part string with conversion already done. The shape class understands how to convert two and from database storage values and forms an interface that can be queried and interpreted.

Each Shape class has a corresponding javascript module typescript class and a c# general use class so that it can be used as a model and be exposed from a backend API to the the front layer. The shape classes are data models that are served from an API controller and we use them for defining routes.

DREAM API Routes					
EntityName	MethodName	Verb	Input	Output	Description
Contact	GetALL	HttpGet	None	List Of Contact	Get all contacts
Contact	Get	HttpGet	ContactId : Int	Contact	Search a contact
Contact	Insert	HttpPut	Contact	ContactId or ErrorLog	Add a contact
Contact	Update	HttpPost	Contact	ContactId or ErrorLog	Edit a contact
Contact	Delete	HttpDelete	ContactId : Int	Boolean	Remove a contact

For a detailed list of all routes please look at Table 1 in the appendix section of this document.

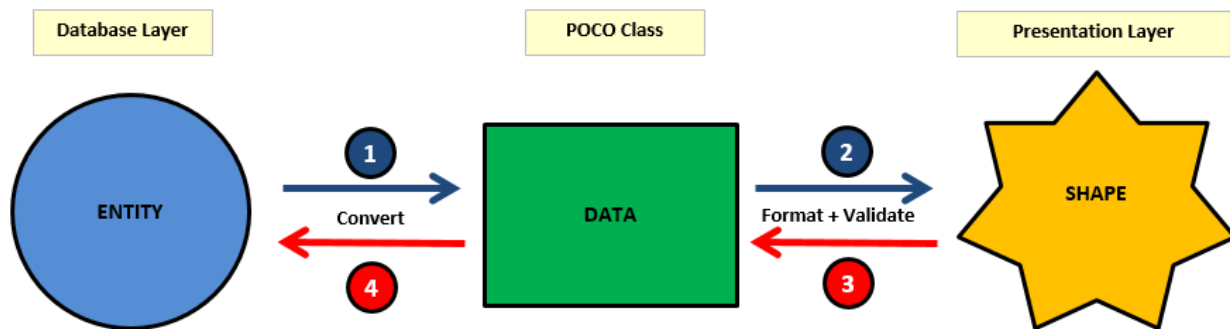
4c – Define Repository Classes :

For every Data Entity class and Data Shape Class there is a corresponding Data Repository class which is the transport mechanism that is used to make the middle tier database calls and hydrate data into the Shape class so that it can be sent over to the UI for binding and use in user screens. The application engine uses the following repository classes:

1. CityRepository – Insert, Update, Delete City data.
2. ContactRepository – Insert, Update, Delete Contact data.
3. ErrorLogRepository – read error log.
4. LoginRepository – setup application security.
5. ProjectRepository – Insert, Update, Delete project data.
6. URLRepository – Insert, Update, Delete urls data.
7. JobRepository – Insert, Update, Delete Job data.
8. JobTypeRepository – read job types.
9. RoleRepository – read role types.
10. StateRepository – read state codes.
11. TitleRepository – read titles.

JobHub - DREAM API & Framework EBook

The Repository classes use the Irepository design pattern which uses GetOne, GetAll, Insert, Update and Delete methods. The architecture works like this. It uses the MVVM design pattern.



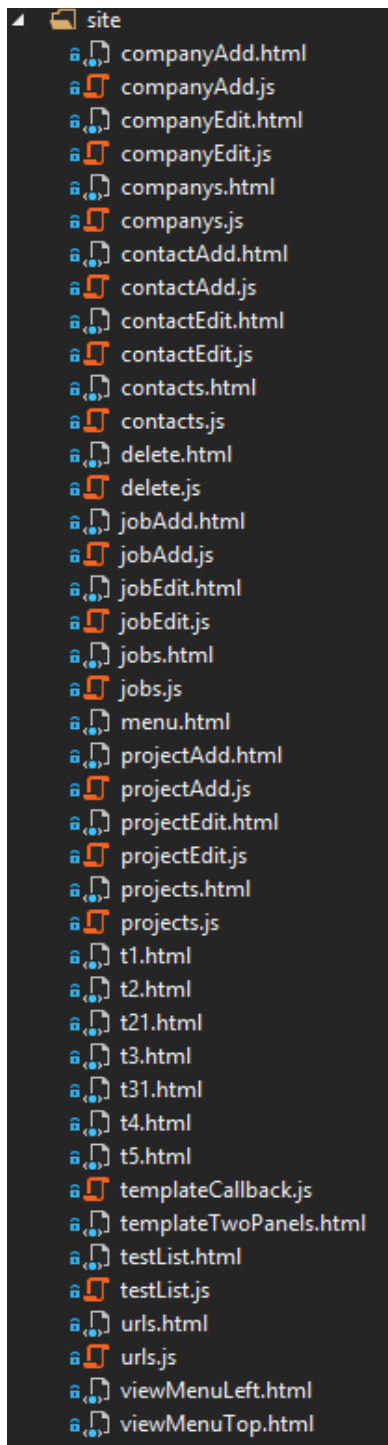
5 – JobHub StoreFront :

The JobHub StoreFront is deliberately built using WCF WebServices because we run into a lot of applications in the real world which are still using WCF and SOAP based webservices so we wanted to show how to migrate these type of services to Web API Version 2 or to the new ASP.NET Core Web APIs. 90% of the code that has been written for the StoreFront has been written in such a way that it will compile in cross platform environments and it will also compile in various types of Visual Studio 2019 projects. The StoreFront was written using Visual Studio 2015 initially.

While the backend and service layer of the StoreFront may still be using WCF, the front end is very very modern and using REACT with requirejs and jquery to build the user interface and this was also done deliberately to show if we just want to upgrade the UI for a legacy application without rewriting its service layer then this is how you would do it.

JobHub - DREAM API & Framework EBook

The screenshot below shows how components are built with REACT javascript. This is one microsite hosted inside the StoreFront. The StoreFront can support multiple REACT based microsites.



JobHub - DREAM API & Framework EBook

The JobHub StoreFront is built with responsive design and will run on a mobile device also. The JobHub StoreFront in action as shown below. Available for demo at <http://improvecode.com>

Improve Code

construct better software;

About Us Dashboard Tools Desktop Web Blog

Contact Info



Location:
Houston, TX USA
Email:
me@improvecode.com
Phone: 1-832-675-3213

Enter email:

REMEMBER ME

We will not send you spam. The email is only used to provide access to articles and downloads.

Access secure content:

LOGIN

Improve code DNA in 7 steps



- I - inception
- M - model
- P - program
- R - define
- O - output
- V - verify
- E - end

Improve code uses a 7 step development **Methodology** which consists of an inception phase to start the project followed by a model phase in which the design and analysis is done. Once the requirements have been gathered, we program the solution and then verify it by doing QA and testing. You reach the end of the development life cycle when there are no more refinements remaining...

Support



Contact us for SQL Server, Oracle, CRM, TFS, ESRI, SharePoint, Biztalk, FileNet, LiveLink & DNN. Expert with server technology.

Industry



Develop custom code for Healthcare, Logistics, Finance, Insurance, Accounting, Energy and Government sector.

Guarantee



Utilize our 7 step process to improve your applications. We create documentation for everything that we deliver.

Guides

- Modern Web Apps: ASP.NET MVC 6 Core Cross Platform & Django. [Read more...](#)
- Desktop Apps: WPF, MVVMLight Toolkit & PRISM. [Read more...](#)
- JavaScript SPA: [HTML5](#), [Angular](#), Aurelia, ExtJS, Durandal, Knockout and other [JavaScript](#) frameworks to build responsive UI. [Read more...](#)
- SharePoint 2013: Stitch together your company Intranet. [Read more...](#)
- EAM Framework: Use Enterprise Asset Management framework to build your IT Landscape. [Read more...](#)

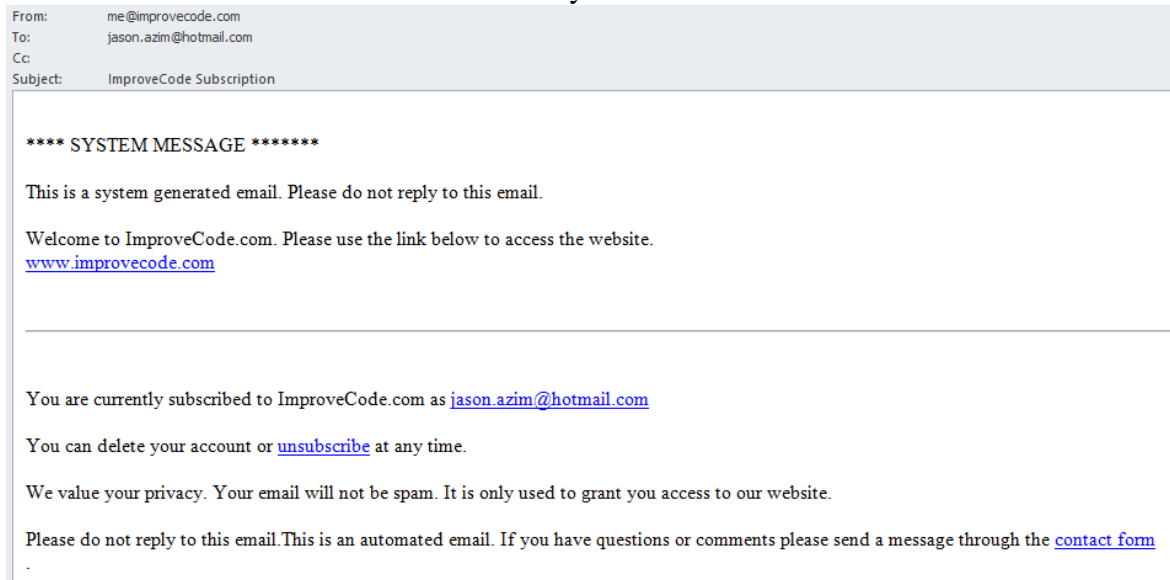
Who uses our products?

Please [view resume](#) for company names and project details. We have 20 years proven track record of well built software.

The StoreFront utilizes a subscription model. If you enter your email and click on the Remember Me button then an email will be generated and you will be subscribed to the website.

JobHub - DREAM API & Framework EBook

The Notification emails look like this when you subscribe.

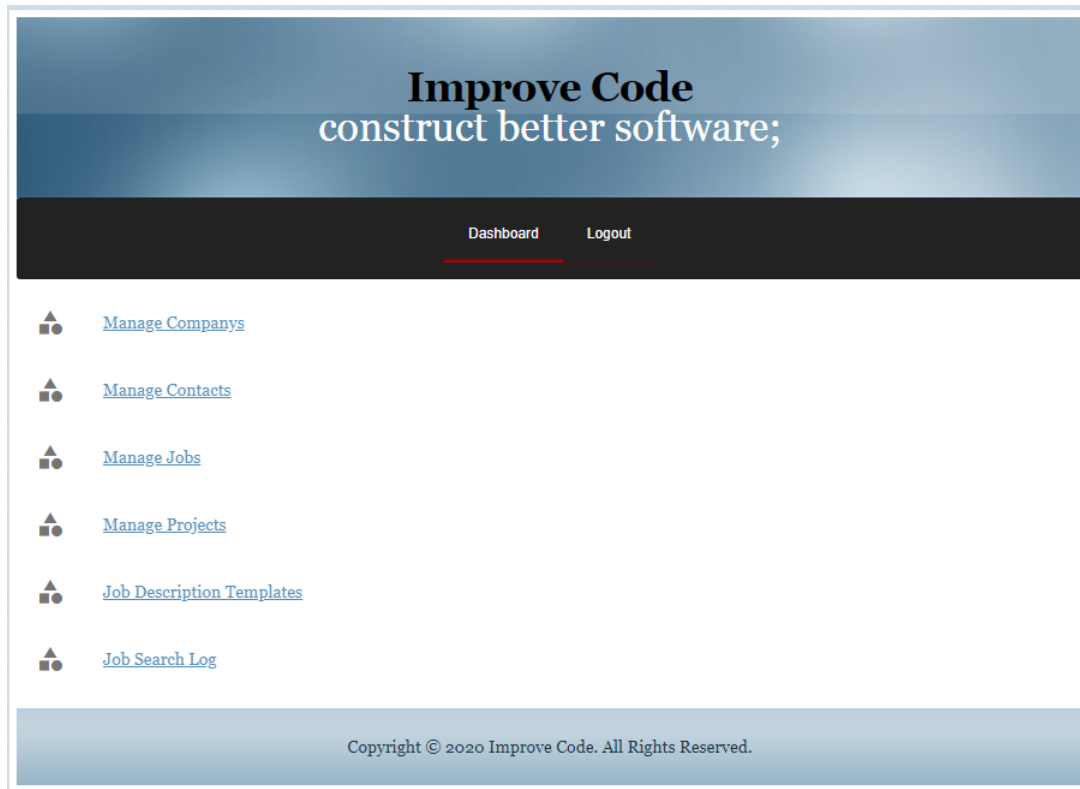


The JobHub StoreFront has controls built with REACT. An example of a search control:
Technology Chart - Areas of focus

Operating Systems	Mac OS Windows Unix Mainframe (IBM ZOS) Linux
Development Environment	Visual Studio Eclipse Xcode Webstorm Intellij
Programming Languages	C# VB.NET Python Java
Source Control	Team Foundation Server JIRA Tortoise SVN
Database	SQL Server Oracle SQL Express SQL Lite
WPF	PRISM MVVM Light Unity MEF Ninject Structure Map WPF Toolkit
Javascript Frameworks	AngularJS ExtJS Wijmo AMD Prototype RequireJS (AMD) Knockout Dojo
SharePoint 2013	WebParts PerformancePoint Infopath SharePoint Designer
ESRI ArcGIS	Web ADF Geocortex
Business Process Manager	Biztalk IBM ODM
Service Bus	Azure IBM Data Power Progress

JobHub - DREAM API & Framework EBook

The JobHub StoreFront Dashboard in action shown below. Built with material design specification Templates. Available for demo at <http://improvecode.com/Dash.aspx>



Clicking on a link displays a list of companys. Then we can edit, delete and update them.

IMPROVECODE HOME COMPANY CONTACT JOB PROJECTS

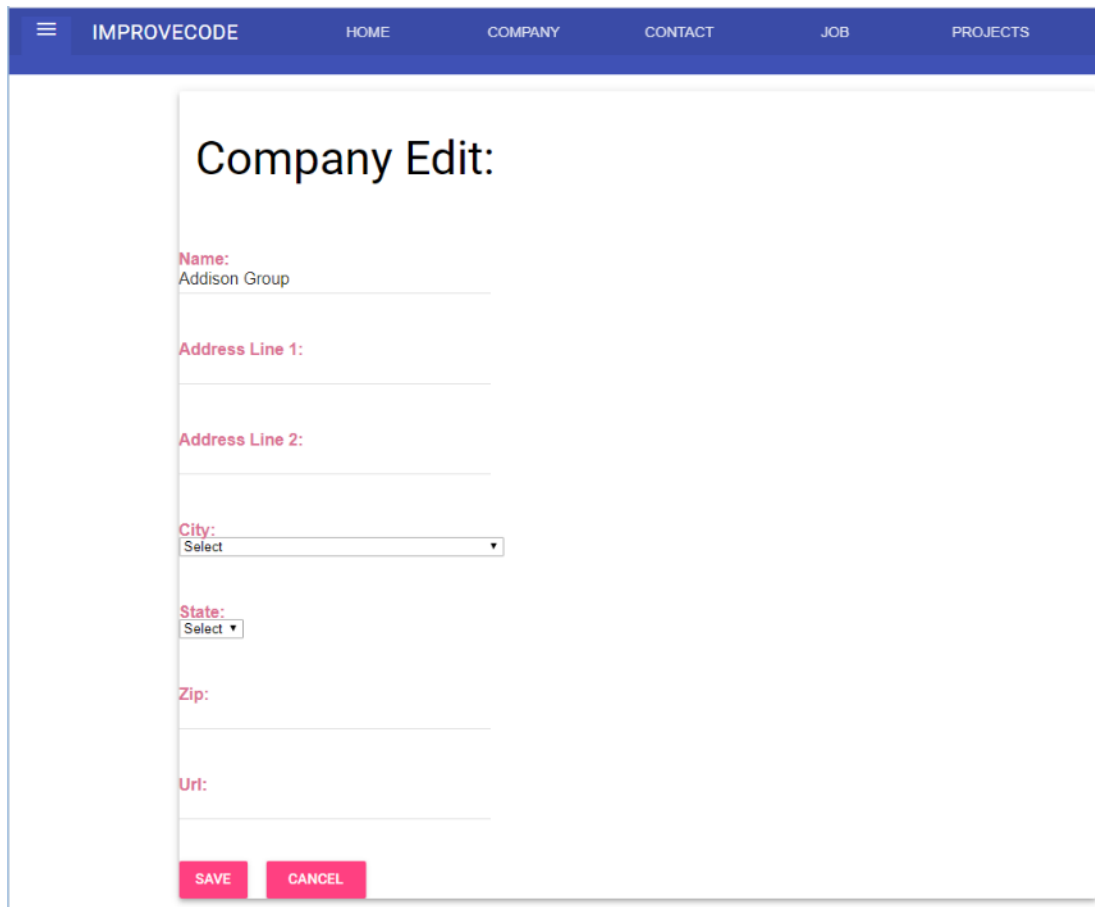
Company List:

ADD REFRESH

Name	AddressLine1	City	State	PostalCode	Uri	Edit	Delete
Addison Group	null	null	null	null	null	Edit	Delete
Adsusman	null	null	null	null	null	Edit	Delete
Advantage Technical Resourcing	null	null	null	null	null	Edit	Delete
AIG	null	null	null	null	null	Edit	Delete
VALIC	null	null	null	null	null	Edit	Delete
Apex Systems	null	null	null	null	null	Edit	Delete
Aquent	null	null	null	null	null	Edit	Delete
HP	null	null	null	null	null	Edit	Delete

JobHub - DREAM API & Framework EBook

Click on the edit link to edit the company which will display the screen shown below.



The screenshot displays the 'Company Edit' form within the JobHub Desktop Admin App. The app's navigation bar is blue and contains the following items: a hamburger menu icon, 'IMPROVECODE', 'HOME', 'COMPANY', 'CONTACT', 'JOB', and 'PROJECTS'. The form itself is white and titled 'Company Edit:'. It contains the following fields:

- Name:** A text input field containing 'Addison Group'.
- Address Line 1:** An empty text input field.
- Address Line 2:** An empty text input field.
- City:** A dropdown menu with 'Select' as the current selection.
- State:** A dropdown menu with 'Select' as the current selection.
- Zip:** An empty text input field.
- Uri:** An empty text input field.

At the bottom of the form, there are two buttons: 'SAVE' and 'CANCEL', both in red.

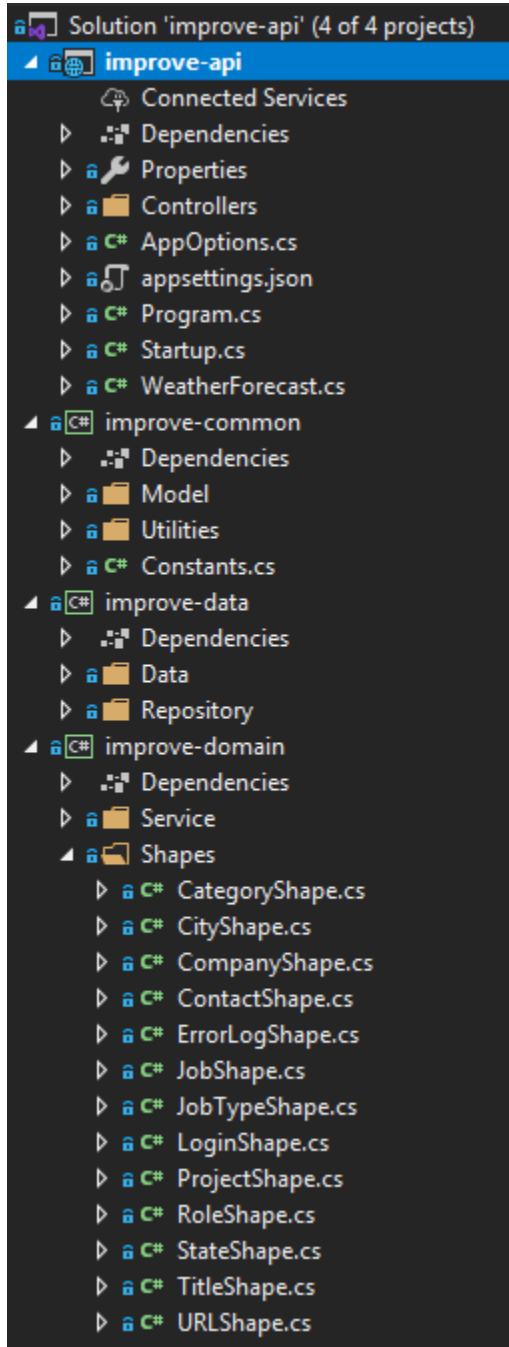
6 – JobHub Desktop Admin App :

Next we will rewrite the StoreFront microsite as an Angular 9 client application. We will build the application in two formats

1. The Web API ASP.NET Core and Angular Node Module Project are housed into one solution. In this setup Visual Studio 2019 is used to run everything. This does not require CORS to be configured because everything is housed into a single project. This architecture is good for production use when we want to deploy all components and everything in one cloud website.
2. The Web API ASP.NET Core is one solution and the Angular Node Module project is a second solution and we use two editors namely Visual Studio 2019 and Visual Studio Code to compile and run the application. This architecture is good for a development or staging environment where we may have multiple developers working on various parts of the application and in this setup the UI can be setup in one GIT repository and the backend Web API can be placed in a separate project.

JobHub - DREAM API & Framework EBook

The ASP.NET Core Web API version of the project is called `improve-api`. It has the architecture shown below:



The **improve-api** project contains the startup classes which are used to run the project and it contains the Controllers which call the repository classes to do all middle tier data retrieval.

The **improve-common** project contains all commonly used infrastructure classes which are shared between the projects and it also contains the Data Models which are mapped to the corresponding database tables.

The **improve-data** project contains all data and repository classes which utilize the IRepository unit of work design pattern and have methods like GetOne, GetALL, Insert, Update and Delete for each entity.

The **improve-domain** project contains all the presentation layer classes which are bound to the User Interface. It also contains the service layer which can be used to setup software as a service for each entity.

JobHub - DREAM API & Framework EBook

The Angular Front End Project Looks like this

```

  services
  TS auth.service.ts
  TS city.service.ts
  TS company.service.ts
  TS contact.service.ts
  TS global.service.ts
  TS http-error-handler.service.ts
  TS job.service.ts
  TS log.service.ts
  TS message.service.ts
  TS project.service.ts
  TS request-cache.service.ts
  TS state.service.ts

```

The **angular-services** folder contains all the services that are used to retrieve data from the backend Web API controllers. There are shared services for authentication, logging, caching and messaging also.

```

  shapes
  TS category.shape.ts
  TS city.shape.ts
  TS company.shape.ts
  TS company.ts
  TS contact.shape.ts
  TS contact.ts
  TS job.shape.ts
  TS job.ts
  TS jobType.shape.ts
  TS project.shape.ts
  TS project.ts
  TS role.shape.ts
  TS state.shape.ts
  TS title.shape.ts

```

The **angular-shapes** folder contains the javascript versions of the C# POCO classes which are used to store and bind the data to angular components.

JobHub - DREAM API & Framework EBook

```
▼ components
  <> companyAdd.component.html
  TS companyAdd.component.ts
  <> companyEdit.component.html
  TS companyEdit.component.ts
  <> companys.component.html
  TS companys.component.ts
  # components.css
  <> contactAdd.component.html
  TS contactAdd.component.ts
  <> contactEdit.component.html
  TS contactEdit.component.ts
  <> contacts.component.html
  TS contacts.component.ts
  <> counter.component.html
  TS counter.component.spec.ts
  TS counter.component.ts
  <> fetch-data.component.html
  TS fetch-data.component.ts
  <> home.component.html
  TS home.component.ts
  <> jobAdd.component.html
  TS jobAdd.component.ts
  <> jobEdit.component.html
  TS jobEdit.component.ts
  <> jobs.component.html
  TS jobs.component.ts
  <> messages.component.html
  TS messages.component.ts
  <> projectAdd.component.html
  TS projectAdd.component.ts
  <> projectEdit.component.html
  TS projectEdit.component.ts
  <> projects.component.html
  TS projects.component.ts
```

The **angular-components** folder contains the html view and the corresponding view model classes which are used to ADD, EDIT and LIST screens for each object model class. There is a shared stylesheet called components.css.

JobHub - DREAM API & Framework EBook

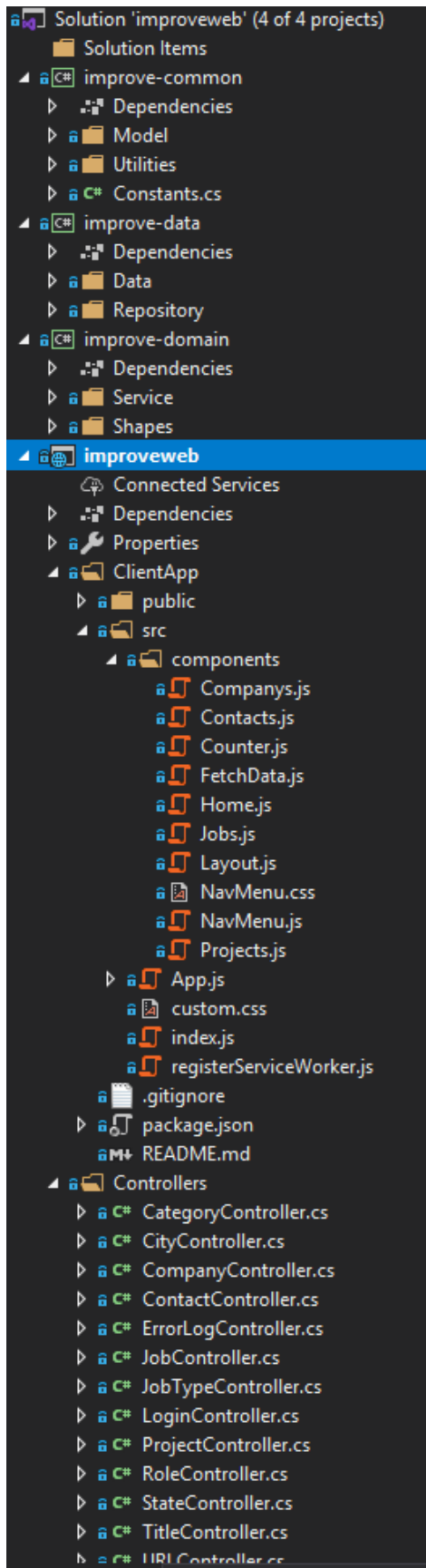
The angular client version 9 is used to compile and run the application.

```
cmd ng serve
'C:\Program' is not recognized as an internal or external command,
operable program or batch file.
C:\global>cd improve-app
C:\global\improve-app>ng serve
10% building 3/3 modules 0 active[HPM] Proxy created: /api -> https://localhost:44374
[HPM] Subscribed to http-proxy events: [ 'error', 'close' ]
chunk {main} main.js, main.js.map (main) 407 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 141 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 6.15 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 9.75 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 3.41 MB [initial] [rendered]
Date: 2020-05-01T17:33:28.045Z - Hash: e954e58b4e2a7357ed99 - Time: 16516ms
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **
: Compiled successfully.
```

6 – JobHub Cloud Service :

Next we will built the application as a cloud service using asp.net web api. Again the same code base can be ported over as a .net standard library to run as a software as a service but this time we will add a azure service bus and we will also add the ability to store json streams to azure blob storage and cosmos db.

JobHub - DREAM API & Framework EBook



The project is setup the same way as before with domain, data, common and shape layers which are all part of the DREAM API and framework. It is built to run in the cloud this time as a Software as a Service.

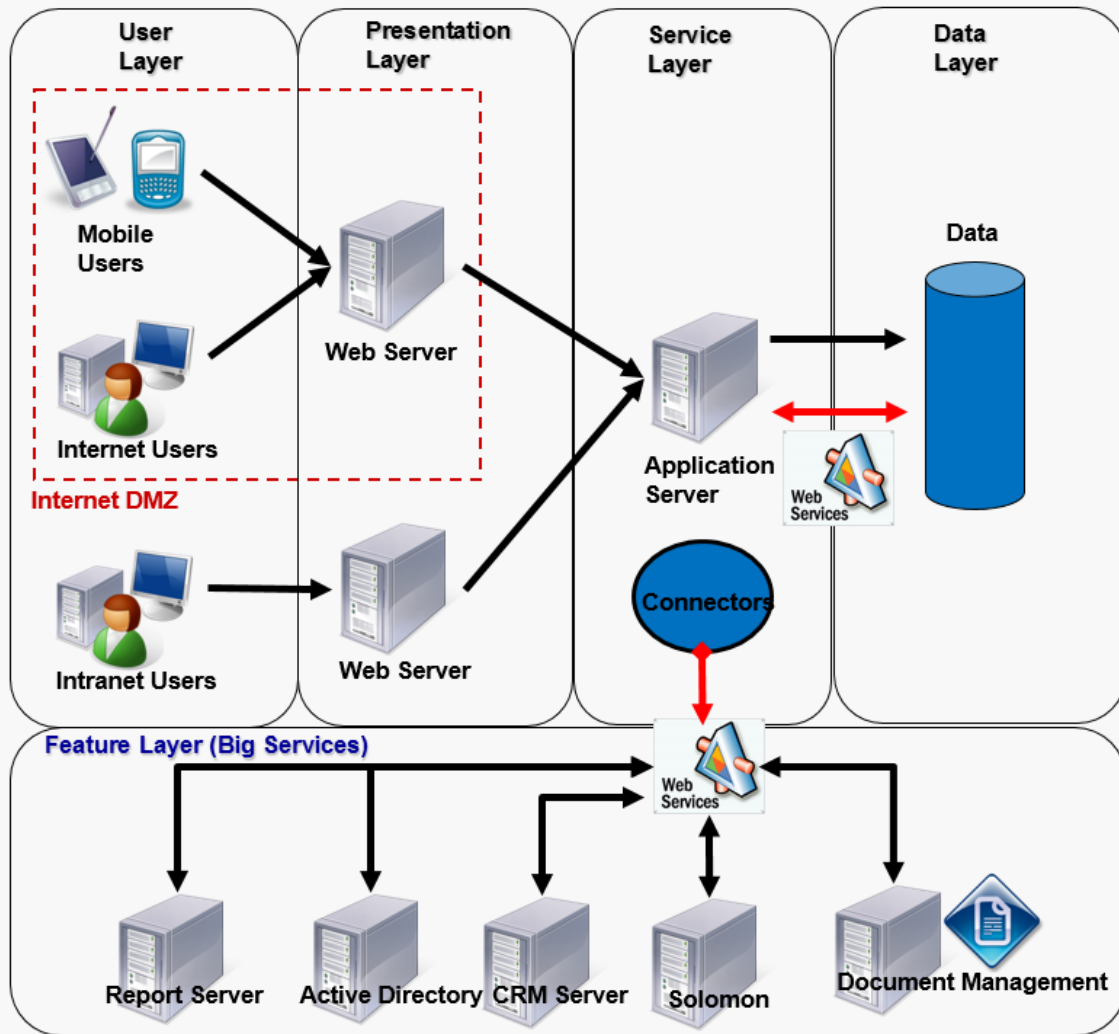
JobHub - DREAM API & Framework EBook

Appendix :

Table 1 - JobHub API Routes



Uri	Method	Description
json/Category	POST	Service at http://improvecode.com/serviceSQL.svc/json/Category
json/Category/{id}	GET	Service at http://improvecode.com/serviceSQL.svc/json/Category/{ID}
json/Category/all	POST	Service at http://improvecode.com/serviceSQL.svc/json/Category/all
json/Category/del/{id}	GET	Service at http://improvecode.com/serviceSQL.svc/json/Category/del/{ID}
json/Company	POST	Service at http://improvecode.com/serviceSQL.svc/json/Company
json/Company/{id}	GET	Service at http://improvecode.com/serviceSQL.svc/json/Company/{ID}
json/Company/all	POST	Service at http://improvecode.com/serviceSQL.svc/json/Company/all
json/Company/del/{id}	GET	Service at http://improvecode.com/serviceSQL.svc/json/Company/del/{ID}
json/Contact	POST	Service at http://improvecode.com/serviceSQL.svc/json/Contact
json/Contact/{id}	GET	Service at http://improvecode.com/serviceSQL.svc/json/Contact/{ID}
json/Contact/all	POST	Service at http://improvecode.com/serviceSQL.svc/json/Contact/all
json/Contact/del/{id}	GET	Service at http://improvecode.com/serviceSQL.svc/json/Contact/del/{ID}
json/GetData/{id}	GET	Service at http://improvecode.com/serviceSQL.svc/json/GetData/{ID}
json/GetSettings	GET	Service at http://improvecode.com/serviceSQL.svc/json/GetSettings
json/Job	GET	Service at http://improvecode.com/serviceSQL.svc/json/Job
json/Job/{id}	POST	Service at http://improvecode.com/serviceSQL.svc/json/Job/{ID}
json/Job/del/{id}	GET	Service at http://improvecode.com/serviceSQL.svc/json/Job/del/{ID}
json/JobType	POST	Service at http://improvecode.com/serviceSQL.svc/json/JobType
json/JobType/{id}	GET	Service at http://improvecode.com/serviceSQL.svc/json/JobType/{ID}
json/JobType/{id}, BodyStyle = WebMessageBodyStyle.WrappedRequest	POST	Service at http://improvecode.com/serviceSQL.svc/json/JobType/{ID}, BodyStyle = WebMessageBodyStyle.WrappedRequest
json/JobType/all	GET	Service at http://improvecode.com/serviceSQL.svc/json/JobType/all
json/JobType/del/{id}	GET	Service at http://improvecode.com/serviceSQL.svc/json/JobType/del/{ID}
json/Log	POST	Service at http://improvecode.com/serviceSQL.svc/json/Log
json/Log/{id}	GET	Service at http://improvecode.com/serviceSQL.svc/json/Log/{ID}
json/Log/all	GET	Service at http://improvecode.com/serviceSQL.svc/json/Log/all
json/Login	GET	Service at http://improvecode.com/serviceSQL.svc/json/Login
json/Login/{id}	POST	Service at http://improvecode.com/serviceSQL.svc/json/Login/{ID}
json/Login/del/{id}	POST	Service at http://improvecode.com/serviceSQL.svc/json/Login/del/{ID}
json/Project	GET	Service at http://improvecode.com/serviceSQL.svc/json/Project
json/Project/{id}	POST	Service at http://improvecode.com/serviceSQL.svc/json/Project/{ID}
json/Project/all	POST	Service at http://improvecode.com/serviceSQL.svc/json/Project/all
json/Project/del/{id}	GET	Service at http://improvecode.com/serviceSQL.svc/json/Project/del/{ID}
json/Role	POST	Service at http://improvecode.com/serviceSQL.svc/json/Role
json/Role/{id}	GET	Service at http://improvecode.com/serviceSQL.svc/json/Role/{ID}
json/Role/all	POST	Service at http://improvecode.com/serviceSQL.svc/json/Role/all
json/Role/del/{id}	GET	Service at http://improvecode.com/serviceSQL.svc/json/Role/del/{ID}
json/Skill	POST	Service at http://improvecode.com/serviceSQL.svc/json/Skill
json/Skill/{id}	GET	Service at http://improvecode.com/serviceSQL.svc/json/Skill/{ID}
json/Skill/all	POST	Service at http://improvecode.com/serviceSQL.svc/json/Skill/all
json/Skill/del/{id}	GET	Service at http://improvecode.com/serviceSQL.svc/json/Skill/del/{ID}
json/Title	POST	Service at http://improvecode.com/serviceSQL.svc/json/Title
json/Title/{id}	GET	Service at http://improvecode.com/serviceSQL.svc/json/Title/{ID}
json/Title/all	POST	Service at http://improvecode.com/serviceSQL.svc/json/Title/all
json/Title/del/{id}	GET	Service at http://improvecode.com/serviceSQL.svc/json/Title/del/{ID}
json/Url	POST	Service at http://improvecode.com/serviceSQL.svc/json/Url
json/Url/{id}	GET	Service at http://improvecode.com/serviceSQL.svc/json/Url/{ID}
json/Url/all	POST	Service at http://improvecode.com/serviceSQL.svc/json/Url/all
json/Url/del/{id}	GET	Service at http://improvecode.com/serviceSQL.svc/json/Url/del/{ID}
json/Url/update	POST	Service at http://improvecode.com/serviceSQL.svc/json/Url/update

Custom Solution - Physical Architecture



JobHub - DREAM API & Framework EBook

Chose a Framework

Mac Development 1	Mac Development 2	Cross Platform 1	Cross Platform 2
Safari & JQuery UI	Google Chrome	Google Chrome	Google Chrome
JSON Rest Kit	JSON Rest Kit	Aurelia or Angular	Generate Native Code
PHP	PHP	Node JS	Xamarin
Cake PHP 	Symfony 	JSON Rest Kit Web API	JSON Rest Kit Web API
Apache Web Server	Apache Web Server	Apache on Mac	Apache
Oracle Database	My SQL Database	Oracle on Mac	Oracle

Windows Development 1	Windows Development 2	Windows Development 3	Windows Development 4
Internet Explorer	Internet Explorer	Internet Explorer	Internet Explorer
DataSet	Aurelia	Angular 2 & Typescript	Django 1.8
WebForms	JSONP	JSONP	JSONP
Enterprise Library	MVC Web API	ASP.NET Core MVC 5	Python (Anaconda)
ODP.NET	Dapper Micro ORM	Entity Framework	Dapper Micro ORM
Oracle	Oracle	Oracle	Oracle