Acadal development playbook

[1.1 Acadal important terminology from a development point of view 2](#_Toc533073348)

[1.2 Acadal web app design system 2](#_Toc533073349)

[1.3 Platforms and development tools 2](#_Toc533073350)

[1.3.1 Password manager – LastPass 3](#_Toc533073351)

[1.3.2 Acadal web app live system 3](#_Toc533073352)

[1.3.3 Acadal web app test system 3](#_Toc533073353)

[1.3.4 Jira 3](#_Toc533073354)

[1.3.5 bitbucket 3](#_Toc533073355)

[1.3.6 sentry 3](#_Toc533073356)

[1.3.7 Html email design – BEEpro 3](#_Toc533073357)

[1.3.8 Email sending service for the webapp – Postmark 4](#_Toc533073358)

[1.3.9 Webapp browser/device testing – browserstack 4](#_Toc533073359)

[1.3.10 Icons and graphics - flaticon 4](#_Toc533073360)

[1.3.11 UX design and documentation 4](#_Toc533073361)

[1.4 Overall system architecture 6](#_Toc533073362)

[1.4.1 The Azure SQL Server 6](#_Toc533073363)

[1.4.2 The C# ASP.NET Web API backend 6](#_Toc533073364)

## Acadal important terminology from a development point of view

The Acadal web app has several user roles and context specific terminology that is important understand to be able work with the platform. There is also a fair amount of technical debt that use outdated terminology and refers to user roles no longer in use. This will not go in depth with the general business model.

User of the Acadal web app can be assigned roles in one or more companies.

* **Customer**: Standard user
* **Consultant**: Access to create sprint templets and module templets
* **Administrator**: Access company settings, team administration and sprint administration, create sprint templets and module templets, access to all sprints in company
* **Super Admin**: The internal system admin used by Acadal staff to create and administrate all companies in the system

For each sprint within company a user can have one of 3 roles.

* **Participant:** Standard user participating in sprint flow and idea rating
* **Sprint Master:** Alsoparticipating in sprint flow and idea rating and access to sprint overview, sprint setup and can create sprint report
* **Consultant:** Can freely navigate all modules, cannot create ideas and access to sprint overview, sprint setup and can create sprint report

Legacy roles and concepts that might be mentioned in the code and other technical debt:

* Co-creator: is the old name for Consultant and there should in most cases be a one-one relation with that functionality.
* Customer-owner: is the old name for Administrator and there should in most cases be a one-one relation with that functionality.
* Co-creator owner, Partner Owner, Partner are roles that is no longer in use and should be omitted.

## Acadal web app design system

The Acadal design web app is mainly based on Material design with an 8dp grid. This means that all UI element sizes and spacings should be dividable by 8. i.e 0,5x: 4dp, 1X: 8dp, 2X: 16 dp, 3X: 24 dp, 6X: 48 dp etc.

There can be some technical debt with 10px and 15px spacing that needs to be converted to the 8dp grid.

<https://material.io/>  
<https://material.angular.io/>

## Platforms and development tools

### Password manager – LastPass

All passwords for the systems/services we use at Acadal should be in last pass. There is separate shared folders for UX, Development, and Support to ensure that only the ones that need assess has it.

<https://www.lastpass.com>  
Ask Morten or Niels for a user

### Acadal web app live system

For security reasons all test users on the live system should have unique passwords generated by LastPass

<https://app.acadal.com>  
Login and Password: ”Use LastPass”

### Acadal web app test system

Most test users on the test system has “cblCBL18” as their password. For security reasons there must not be any customer data on the test system.

https://latest.acadal.com  
Login and Password: ”Use LastPass”

### Jira

We use Jira as our platform for all task management on the webapp. Epics are used as a container for the major capability’s of the systems. Each version of the capability gets a number e.g. *Notifications 1*, *Notifications 2*, *Notifications 3* etc. Only functionality/storrys/bugs that is a priority in the current short-term plan is a signed to a numbered epic all other issues it put in a roadmap epic e.g. *Notifications Roadmap.*

We use storrys for all new functionality and bugs for functionality/issues discovered not work as intended.

Our main project is called Acadal Products and we use a Kanban board for work planning.

<https://acadal.atlassian.net>  
Admin login: [engineering@acadal.com](mailto:engineering@acadal.com)  
Password: ”Use LastPass”

### bitbucket

https://bitbucket.org/  
Admin login: [engineering@acadal.com](mailto:engineering@acadal.com)  
Password: ”Use LastPass”

### sentry

<https://sentry.io/auth/login/>  
Admin login: [engineering@acadal.com](mailto:engineering@acadal.com)  
Password: ”Use LastPass”

### Html email design – BEEpro

We use a html email designer called BeePro to create the look and feel of the emails send form the web app. The templets are exported as html and then used as a basis for the templets used in postmark that is our email sending service. Depending on the dynamic content of the mails the html is adjusted/edited in postmark.

Note: All graphics used in the Web-app emails is hosted by BEEpro CDN and that will stop working if the account is canceled.

<https://pro.beefree.io/login>  
[engineering@acadal.com](mailto:engineering@acadal.com)  
Password: ”Use LastPass”

### Email sending service for the webapp – Postmark

<https://postmarkapp.com/>  
Owner: engineering@acadal.com  
Password: ”Use LastPass”

### Webapp browser/device testing – browserstack

Platform to help test the web app on most common browsers and mobile devices. After you have logged in to Browserstack you can open a virtual browser to test if there are any issues using the Acadal web app on that platform.

<https://www.browserstack.com/users/sign_in>  
[browserstack@acadal.com](mailto:browserstack@acadal.com)  
Password: ”Use LastPass”

### Icons and graphics - flaticon

We have a flaticon account where we finde most of icons/graphics user on hubspot/webapp/emails.

<https://www.flaticon.com/profile/login>  
[Acadal@acadal.com](mailto:Acadal@acadal.com)  
Password: ”Use LastPass”

### UX design and documentation

#### Sketch

<https://www.sketchapp.com/>  
LICENSE KEY: SK3-9741-8284-0132-0006-0726  
UPDATES AVAILABLE UNTIL: Jun 26, 2019

All sketch files can be found in Acadal Onedrive: Team Acadal - Documents\General\UX\Sketch filer

#### Axure

Axure account for Axure share with team repository:

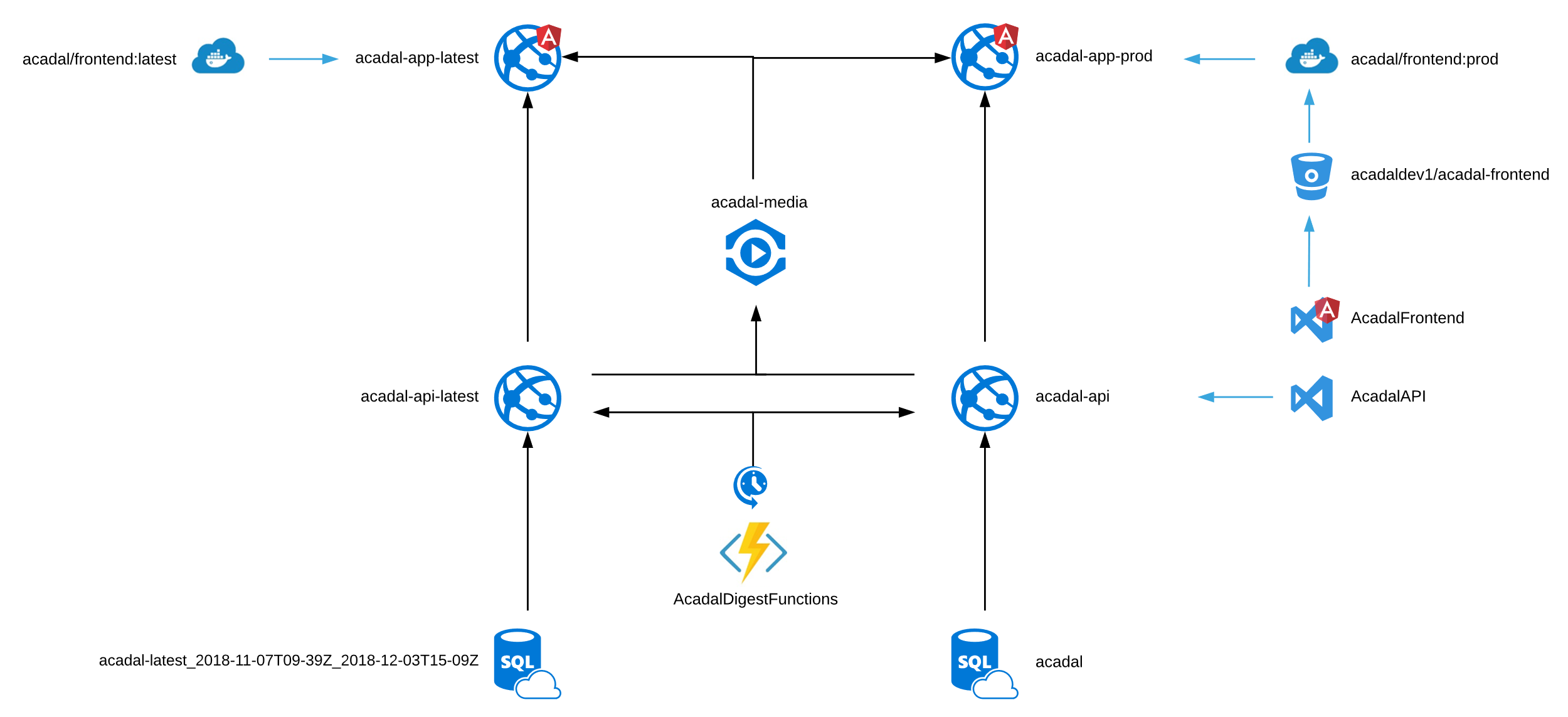
URL: <https://share.axure.com>  
User: [ux@acadal.com](mailto:ux@acadal.com)  
Password: ”Use LastPass”  
Project name: Acadal Wireframes

External access to html export of Acadal Wireframes  
Url: <https://kw6urb.axshare.com/>  
Password: cblCBL18

Most designs are in folders according to Jira epic and if a design refers to a Jira story this is added to the page name. There is also a folder containing general “System diagrams” and a folder with Archived designs that are done or out of date.

## Overall system architecture

The system consists of an Azure SQL Server, an Azure Web App running a C# ASP.NET Web API backend, and an Azure Web App running a containerized Angular 7 frontend.



### The Azure SQL Server

Aptly named **acadal-sql**, the server currently hosts 2 SQL databases. One named a**cadal** which contains all data for our production environment, and one (currently) named **acadal-latest\_2018-11-07T09-39Z\_2018-12-03T15-09Z** which contains all data for our test environment.

Connection Strings for each can be found in the Web.config file in the **AcadalAPI** Visual Studio project.

We replace the database for our test environment whenever necessary, something that is reflected in its unnecessarily complex name, which was auto-generated by Azure when the database was last copied.

We routinely connect to this server through Microsoft SQL Server Management Studio whenever we need to monitor the data in a database, or when we need to manually run queries, etc.

### The C# ASP.NET Web API backend

The backend is a monolithic [big ball of mud](https://en.wikipedia.org/wiki/Big_ball_of_mud), with no clear boundaries between business domains. All system entities are fused together. Very little logic is covered by unit tests, with close to nothing even unit testable. Code duplication is rampant, and there is considerable technical debt.

The code base is currently being refactored from having all logic fused directly into the controllers, to having thin controllers that instead use unit testable [coordinators](https://en.wikipedia.org/wiki/Mediator_pattern) referencing [repositories inside units of work](https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application).

The backend uses [Microsoft Entity Framework 6,](https://docs.microsoft.com/da-dk/ef/ef6/index) in a code-first configuration. Currently, there are 28 different types of Entities, of which 21 are in use. Relations between these 28 entities are mostly a crisscross spiderweb of cross-referencing class fields, which are automagically converted into badly named foreign keys and indices in SQL tables, not to mention the auto-generated tables handling each existing many-to-many relationship between the entities. It is a mess, but thankfully, a mess that is relatively easy to read. Picking it apart, though, will be another matter.

The backend is manually built and published directly to Azure from Visual Studio. First, a given build is manually built and published to the **acadal-api-latest** Azure Web App. After manually smoke testing that deploy by clicking around on latest.acadal.com, the same build is manually built and published to the to the **acadal-api-staging** deployment slot in the **acadal-api** Azure Web App**.** If deployment ran smoothly, the **acadal-api-staging** deployment slot is swapped with the **acadal-api** production deployment slot through the Azure portal.  
  
This approach will obviously have to be replaced immediately when the development team grows.

### The Angular 7 frontend

The frontend is a standard Angular 7 web app, that uses a relatively small set of 3rd party dependencies. Again, we are dealing with a [big ball of mud](https://en.wikipedia.org/wiki/Big_ball_of_mud), with no clear boundaries between business domains.

Considerable effort has now been put into converting nested service subscription scopes to [RxJS](https://rxjs-dev.firebaseapp.com/)-driven reactive flows. This effort is currently nowhere near completion.

Additionally, the file and folder structure is being improved to a point where it follows [Angular recommendations](https://angular.io/guide/styleguide), and to where [it will scale](https://itnext.io/choosing-a-highly-scalable-folder-structure-in-angular-d987de65ec7). Currently, though, there is much more work to be done.

There were very few reusable components in the frontend at first, but a component library has recently been added that follows the [Atomic Design Methodology](http://atomicdesign.bradfrost.com/chapter-2/). It’s components are now used throughout the frontend, mainly for rendering representations of users. The open source [Angular Material](https://material.angular.io/) component library is used heavily throughout the code base as well.

At one point an effort was made to move our own component library into its own project (**acadal-component-library**), but this effort ultimately failed. The code for that component library project, as well as the code for a storybook project related to the component library (**acadal-storybook**), and a project that ran e2e tests for the storybook project (**acadal- storybook-e2e**), remains.

The end-to-end project for the main app project (**acadal-app-e2e**) runs relatively well but will fail when too much data exists in the test environment database, when more than 25 Companies exist, and often for no apparent reason as well. This is primarily caused by the fact that the frontend is configured to use **acadal-api-latest** backend when running end-to-end tests. This is deliberate, and very useful right now, but less than ideal once more developers need to collaborate.

Read the README.md for the **AcadalFrontend** project to learn more about how Continuous Delivery is set up for the frontend.