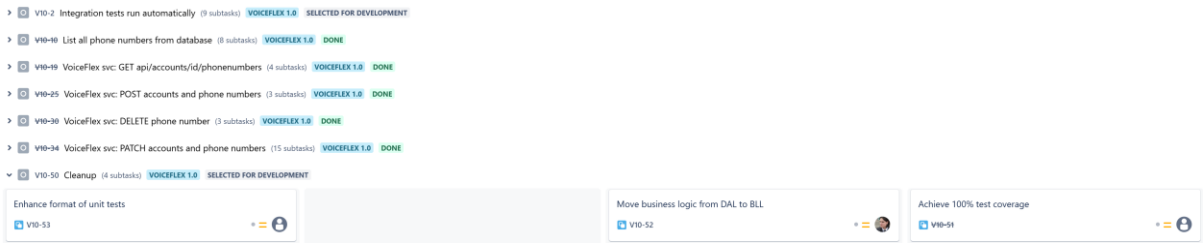
Daniel Schoder – Technical Test Approach

Jira project Kanban board

I started by creating a new JIRA project with a Kanban board, one epic and a first story. Then I broke this story down into sub-tasks and used them to create the foundations.



VoiceFlexTest Service

Applying a radical TDD approach I started by creating a Web Service calling the endpoints of the actual service I was tasked to develop (although this actual service did not exist yet). These calls failed of course, but they gave me a starting point.

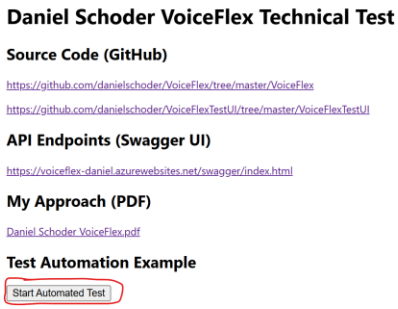
VoiceFlex Service

Now I created a .net 7.0 C# Azure Web Service which was intended to act as a backend (microservice) handling the accounts and phonenumbers as described in the technical test requirements.

VoiceFlexTestUI Service

I also created a typescript Azure Static Web App as a front end running a simple automated integration test.

https://voiceflex.schoder.uk/



Azure App Service

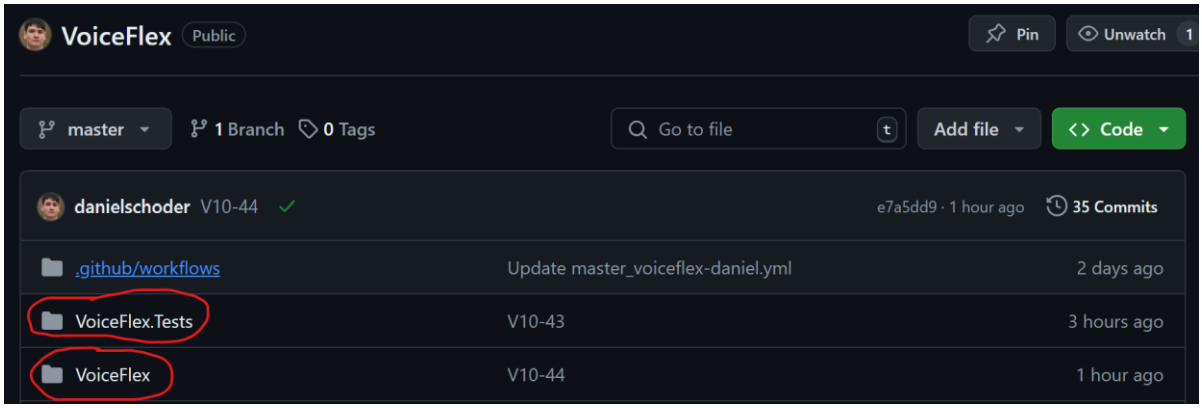
I created a new Azure Web Service to have my VoiceFlex service online and published my first version with one “alive” endpoint.

https://voiceflex-daniel.azurewebsites.net/api

GitHub repository

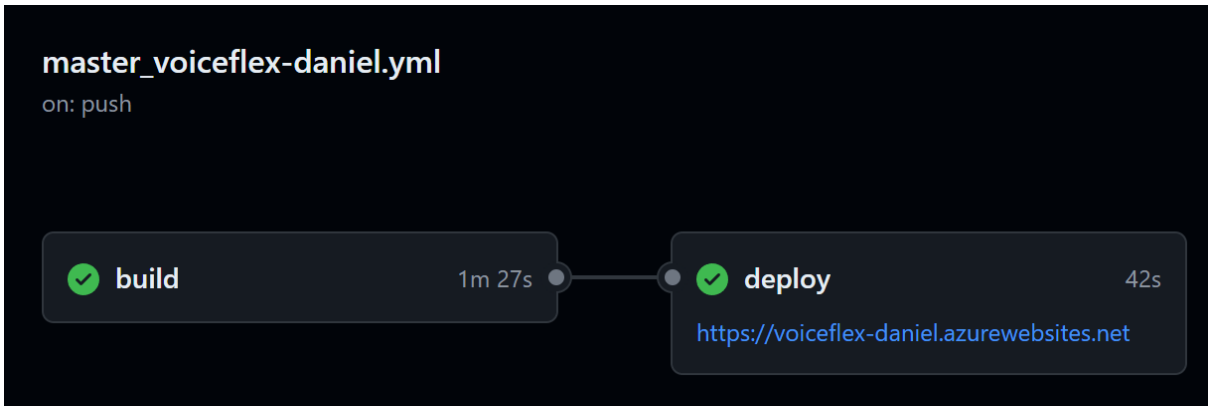
I created a GitHub repository with a master branch.

https://github.com/danielschoder/VoiceFlex



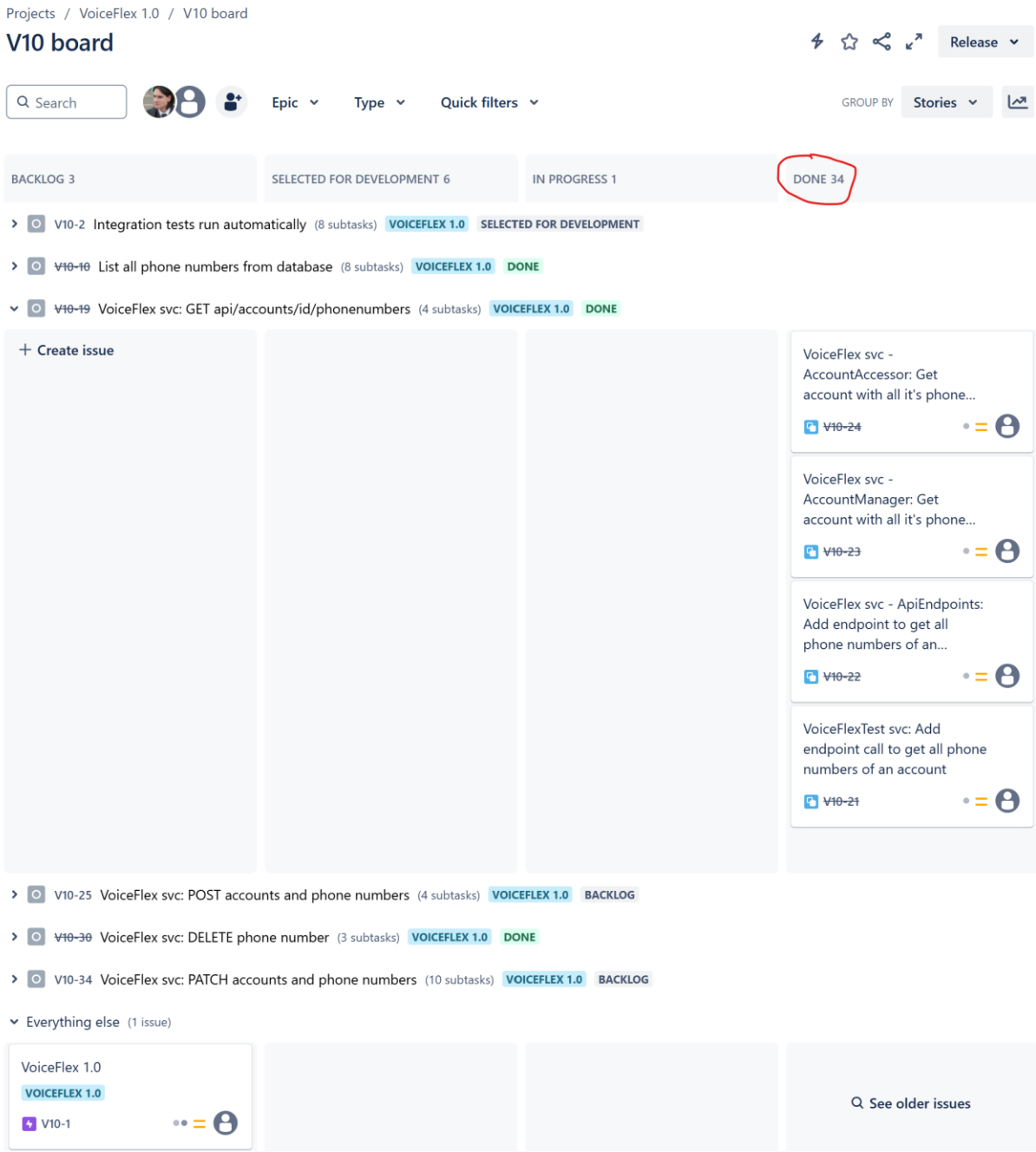
CI/CD pipeline

I create a .yml pipeline to make sure that all my merges into the master branch on GitHub are immediately released. Over the last three days I released new functionality more than 20 times.



Epic, stories, subtasks

I tried to narrow down each story to only a few tiny subtasks and to finish the stories in a way that they offered working functionality to the user immediately when released. Here is an example:



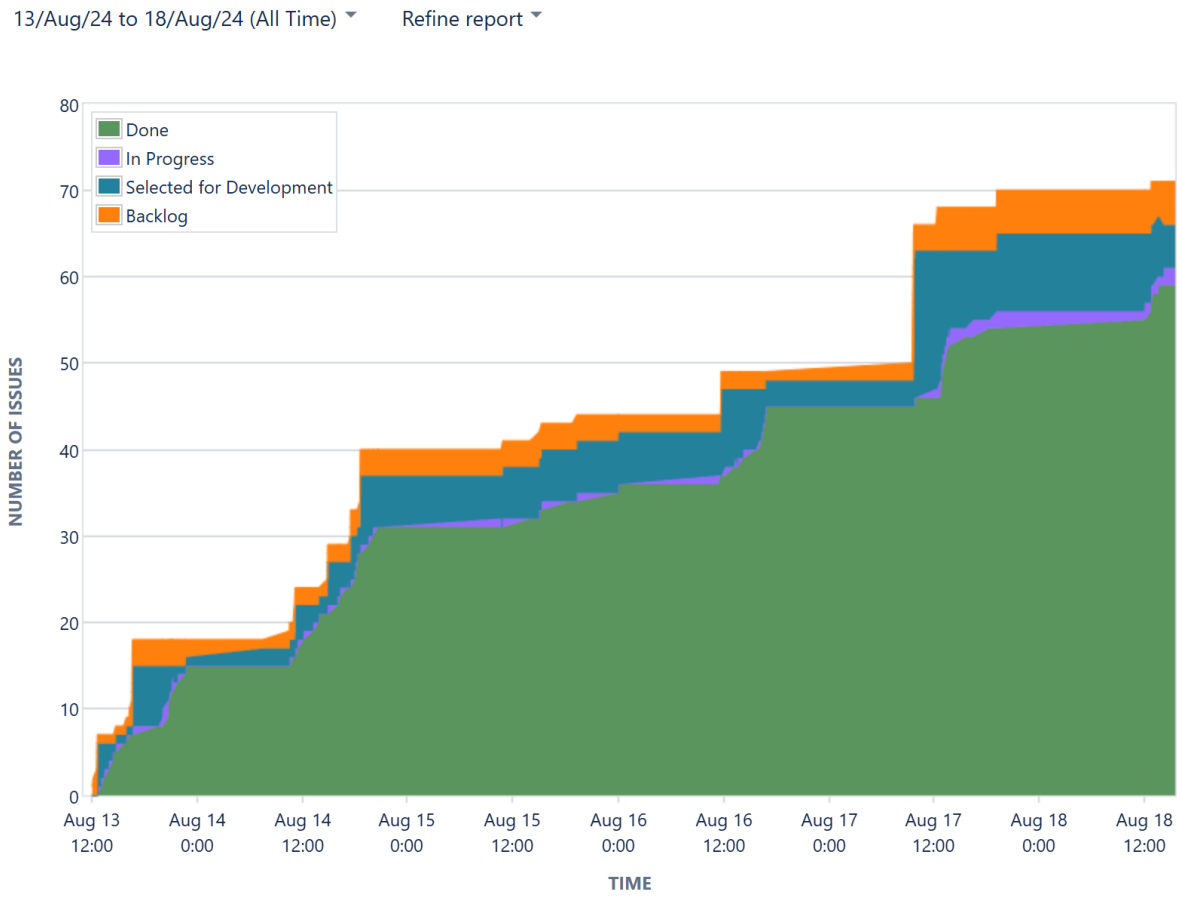
TDD

I found it most efficient to consistently write the unit tests and then developing the code against them until the tests worked. When I started on Tuesday morning, I only had experience with MS Test. Knowing that VoiceFlex uses NUnit I decided to use it here. I also used the advice of ChatGPT to get up to speed with NUnit quickly.

Five days

I got into a flow of creating stories, deriving sub-tasks, writing unit tests, developing code, merging source code into the master branch on GitHub, deploying functionalty and moving tickets across the Kanban board.

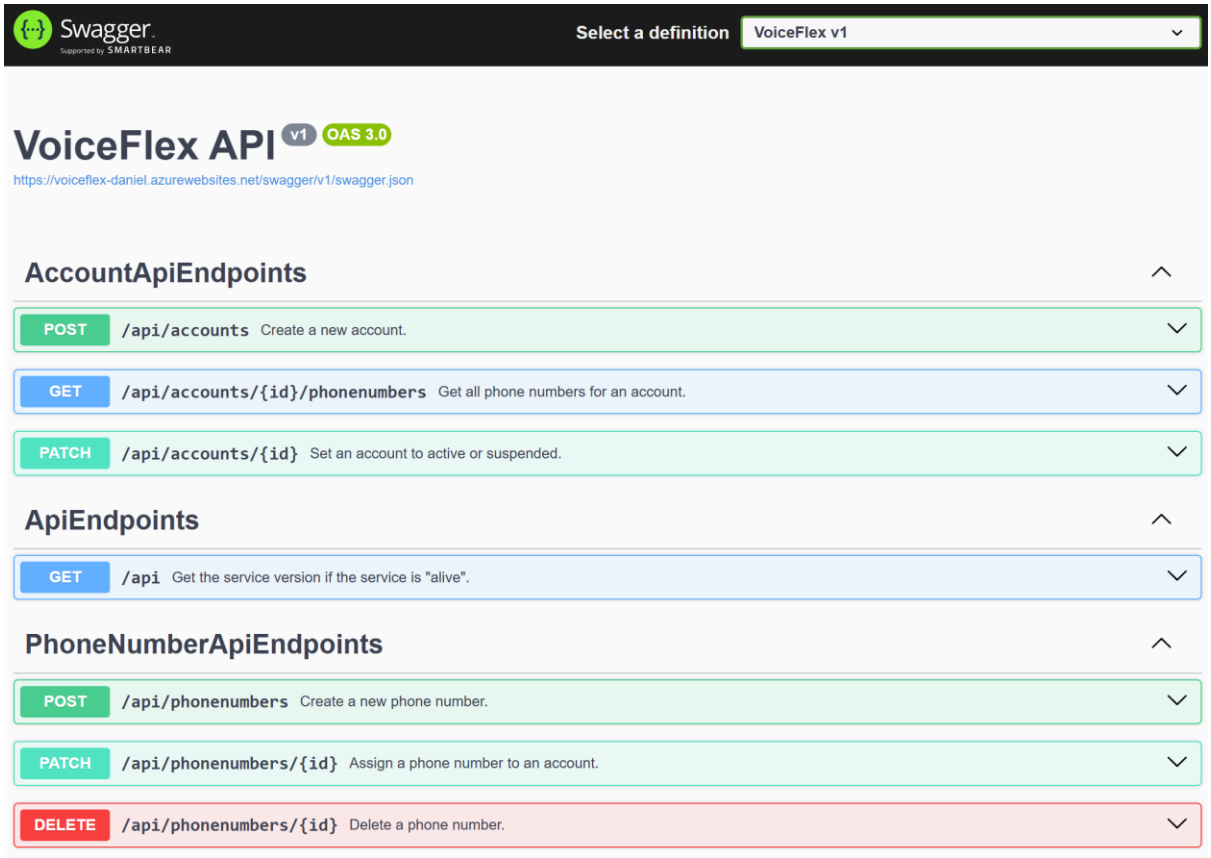
This Jira report reflects my progress over five days:

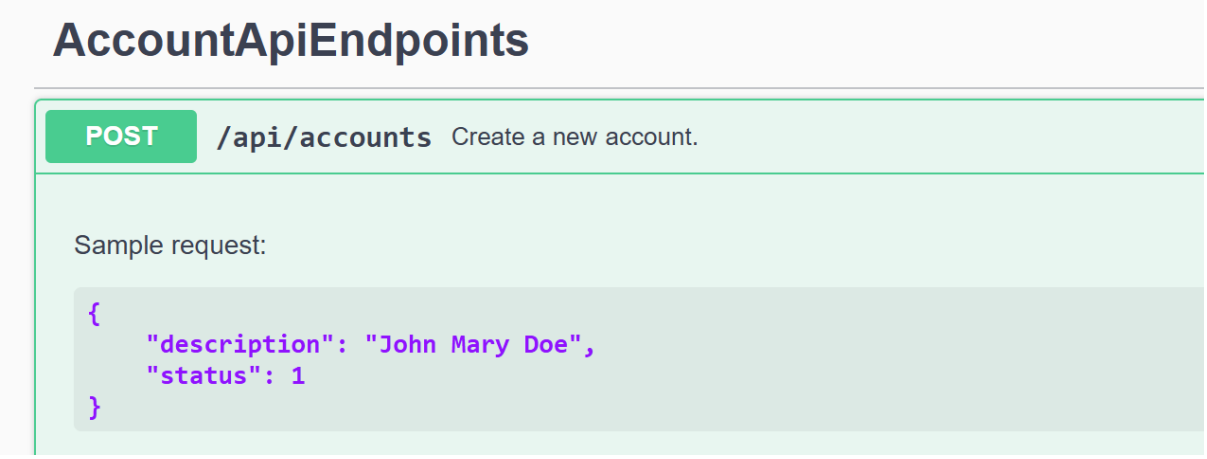


Swagger UI

I made sure that my VoiceFlex service is fully useable via a Swagger UI. I also added a little bit of endpoint documentation there which describes how to use the endpoints.

https://voiceflex-daniel.azurewebsites.net/swagger/index.html





Error catalog

I also made sure that my web service has an internal error catalog with unique error codes. The endpoints return an error JSON including these error codes and error messages when an endpoint detects any error in the request data or cannot process it.

Curl

```
curl -X 'GET' \
  'https://voiceflex-daniel.azurewebsites.net/api/accounts/7e828540-960f-4b3d-1ef5-08dcb3bb26c0/phonenumbers' \
  -H 'accept: */*'
```

Request URL

```
https://voiceflex-daniel.azurewebsites.net/api/accounts/7e828540-960f-4b3d-1ef5-08dcb3bb26c0/phonenumbers
```

Server response

Code	Details
404 *Undocumented*	Error: response status is 404

Response body

```
{
  "code": "VOICEFLEX_0001",
  "message": "A resource with this id could not be found."
}
```

Test coverage

I managed to measure the test coverage of my code and enhanced it from about 90%

Program								
VoiceFlex.ApiEndpoints.AccountApiEndpoints	9	0	9	54	100%	0	0	
VoiceFlex.ApiEndpoints.ApiEndpoints	8	1	9	25	88.8%	0	0	
VoiceFlex.ApiEndpoints.PhoneNumberApiEndpoints	9	0	9	53	100%	0	0	
VoiceFlex.BLL.AccountManager	13	2	15	41	86.6%	3	4	75%
VoiceFlex.BLL.AccountValidator	9	0	9	23	100%	5	6	83.3%
VoiceFlex.BLL.ErrorManager	15	0	15	43	100%	2	2	100%
VoiceFlex.BLL.PhoneNumberManager	24	3	27	56	88.8%	6	8	75%
VoiceFlex.BLL.PhoneNumberValidator	9	0	9	23	100%	5	6	83.3%
VoiceFlex.DAL.AccountAccessor	44	0	44	75	100%	8	8	100%
VoiceFlex.DAL.PhoneNumberAccessor	34	4	38	72	89.4%	8	10	80%

to 100%:

Program								
VoiceFlex.ApiEndpoints.AccountApiEndpoints	9	0	9	54	100%	0	0	
VoiceFlex.ApiEndpoints.ApiEndpoints	9	0	9	25	100%	0	0	
VoiceFlex.ApiEndpoints.PhoneNumberApiEndpoints	9	0	9	53	100%	0	0	
VoiceFlex.BLL.AccountManager	15	0	15	41	100%	4	4	100%
VoiceFlex.BLL.AccountValidator	9	0	9	23	100%	6	6	100%
VoiceFlex.BLL.ErrorManager	15	0	15	43	100%	2	2	100%
VoiceFlex.BLL.PhoneNumberManager	27	0	27	56	100%	8	8	100%
VoiceFlex.BLL.PhoneNumberValidator	9	0	9	23	100%	6	6	100%
VoiceFlex.DAL.AccountAccessor	44	0	44	75	100%	8	8	100%
VoiceFlex.DAL.PhoneNumberAccessor	38	0	38	72	100%	10	10	100%

Code Complexity

Eventually I also managed to keep the code complexity on low levels. Here some examples for the most complex classes in my source code:

Crap Score	Cyclomatic complexity
1	1
1	1
1	1
2	2
1	1
1	1
1	1

Crap Score	Cyclomatic complexity
1	1
6	2
6	2
6	2
6	2
2	2

Crap Score	Cyclomatic complexity
1	1
2	2
2	2
4	4

Assumptions

My solution is based on the following assumptions. In case any of these is do not apply I need to adapt my solution accordingly. I think it is important to be transparent about my assumptions:

1. A new account can be active or suspended. The status is provided in the post JSON.
2. An account description can be duplicate.
3. Phone numbers cannot be duplicate.
4. RESTful endpoints should by defintion be idempotent. Therefore, I implemented the requested “toggle”-functionality as a simple status update (PATCH) which does not change the status back and forth every time it is called.
5. For the same reason I allowed to assign a phone number to the exact person to which it is already assigned – but not to someone else – without resulting in an error.
6. I only implemented the exact endpoints which were defined in the requirements (apart from the “service alive” endpoint). Hence, there is e.g. no list of accounts or of all phone numbers.
7. There was a requirement for a list of phone numbers for a specific account. I implemented that in the format of “api/accounts/{accountid}/phonenumbers” trying to follow the RESTful API standards as described here: https://restfulapi.net/resource-naming/
8. Setting an account to suspended automatically unassigns all phone numbers which are currently assigned to this account.