Console to SAAS

This book will guide you step-by-step how you can build a scalable product from a proof of concept (https://en.wikipedia.org/wiki/Proof_of_concept) to a production ready SAAS (https://en.wikipedia.org/wiki/Software_as_a_service).
Any development done will start from a business need: this will make things clear for the team what is the impact of the delivery.
You will see different arhitecture patterns for separation of concerns (https://en.wikipedia.org/wiki/Separation_of_concerns) and why some of them fit well and some of them not. Everything will happen incrementally and the product development history will be easy to be seen by analyzing commits.
To have our finished product, we will see the two development directions:

On premise solution

- how the idea get a shape
- creating the PoC
- bug fixing
- componentization / testing / refactoring
- unit and integration testing
- extending functionalities

Scaling

- authentication
- load testing
- profiling and optimisations
- continuous deployment
- tight coupled microservices
- messaging systems / loose coupled microservices
- third party integrations

The boss that is hurrying up: proof of concept + source code

True story: one day my boss came and he asked me to have a discussion about how can we help our colleagues from the financial department since they are thinking about automation process. We found a spare place and scheduled a meeting where we find out that 3 employees were manually extracting the client information (name, identity information, address) and contract agreement details (fee, payment schedule) to an Excel file and saving the document in a protected area.

Technical analysis

After putting some brainstorming and analyzing the constraint we ended up developing a solution that reads from a folder share and continue the process without any human intervention. This would allow to have all our employees to continue their work simultaneously with less effort than

before. The prototype solution that we developed was a console app written in C# where we were able to fulfill our requirements in a minimum amount of time.

Exercise

Make a Console solution that reads all word documents from a folder , parses and outputs the contents to Excel.. Create and make it work, the easy way. You can find sample data in the folder data in the github project .

Impact to the solution

We decided to go with a third-party tool (NPOI) instead of us making a library that parses word documents.

Code

```
string folderWithWordDocs = Console.ReadLine();

//omitted code for clarity
foreach (string file in Directory.GetFiles(folderWithWordDocs, "*.docx"))
{
    //omitted code for clarity
    var contractorDetails = ExactContractorDetails(document.Tables[0]);
    allContractors.Add(contractorDetails);
}
```

As you can see, we were facing resolving the solution for the particular client for the specific situation without having an overview of the overall process or any thoughts to going this idea further. The whole process is a sequence of operations without any control flow involved.

Technical Reading Box

Read about Gui.cs - https://github.com/migueldeicaza/gui.cs

Read about the difference between EnumerateFiles and GetFiles.

Read about Async Enumerable in .NET 3.0

Financial department last tweaks: MVP + fixing bugs, source control

We installed our console app on one of the computers of an employee in the financial department, we set it up and we let it running. After a few days we had some bug fixing and some new requests that came in. We saw an enthusiastic behavior on our colleagues by using our solution and that was our first real feedback that we were resolving a customer pain. We anticipated that we will get more feedback and to track what we have done we needed a source control. After a few iterations, we had our financial department happy and we are ready with our first MVP.

Technical analysis

We decided to have 2 separate projects (a business logic and console) because it is not taking so much time and can be beneficial in the long run.
Also, because of the modifications , we need to have a source control - or, at least, to can go back to some version.

Exercise

Try to refactor the solution from Chapter 1 to extract the business logic(i.e. transforming word files into an excel file) into a separate class.

Code

```
var wordExtractor = new WordContractExtractor(folderWithWordDocs);
wordExtractor.ExtractToFile(excelResultsFile);
```

Now, we are ready to process any folder very easy, just be instantiating a new class with the directory

Further reading

Version Control : https://en.wikipedia.org/wiki/Version_control

Set also a repository at http://github.com/ , https://azure.microsoft.com/en-us/services/devops/ or other online source control.

Refactoring: https://www.amazon.com/Refactoring-Improving-Design-Existing-Code/dp/0201485672