Politecnico di Torino

Master's Degree in Computer Engineering

A.a. 2020/2021

July 2021

Improving company communication for remote working: a use case

Supervisors:
prof. Luigi DE RUSSIS
eng. Marcello CERRI

Candidate:
Andrea SCIBETTA

Table of contents

List of figures

Chapter 1

Introduction

1.1 The challenge of COVID-19

This last year represented a new and difficult challenge for many people and companies. With the advent of COVID-19 and many lockdown measures implemented by different countries all over the world, the progress of digitization of the whole society became a fundamental topic. Many different contexts of everyday life were affected by the forced distance. Suddenly everyone was stuck at home, from students to workers, both from public and private agencies. Every aspect of society needed to be reformed and adapted to this new and unique situation where everybody had to stay at home. To keep going on with everyday life, Internet communication and the platforms that provides specific services in that field were of crucial importance.

In this context, the discussions about working from home (WFH) became of crucial importance. Many companies needed immediate and comfortable communication between colleagues and between different areas of the same industry, to keep working at the same pace, trying to avoid the social distance from becoming also mental distance. The use of platforms to ease communication was already growing, but since the stroke of COVID-19 the growth became an explosion. And this led to a massive use of this kind of services, not only for the exchange of messages inside the same working unit, but also from one department to another, for example to speed up communication between human resources and the other company employees.

To allow companies to work remotely a platform for chat and video communication is needed. In this complex situation, where the whole society was forced to remote solutions, this kind of software assumed great importance. In many activities chat platforms were already adopted and used for daily communication, but during the long periods of working from home they became a constant in the workday of the employees.

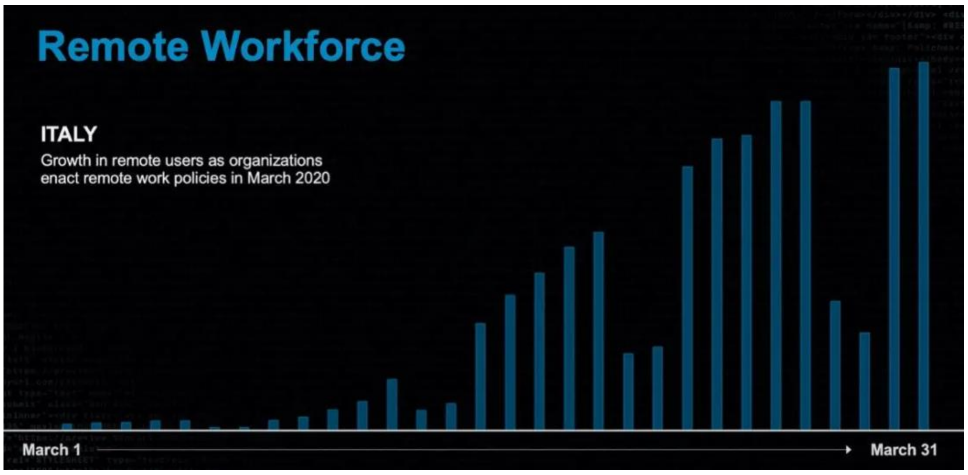


Figure 1.1 The explosion of remote work in Italy during March 2020. Image taken from [12]

The videoconference platforms were something already growing, and the massive adoption of this kind of technology was predictable since at least five years ago [1]. What was not predictable is the pandemic that struck the world during 2020 and is still one of the biggest problems of today's society.

1.2 The importance of communication platforms

This situation became fertile ground for platform like Microsoft Teams, Slack, Zoom and Google Meet to grow even more, and to become the center of coordination and everyday life in many different fields. In this sense, the potential of this type of platforms had already been partially explored. For example, there was great interest in how chatbots create value [8], considering the possibilities to link social interactions within instant messengers, using third-party systems and business processes.

Interesting results were also achieved integrating search engine in Slack [9] by using the Slack conversation API. In this way it is possible to avoid that the user has to switch many platforms, working in a central hub that can provide everything to the worker. Also Zoom had an important role in providing a platform for remote work and in particular remote classrooms. It was particularly used for online courses, exploiting the asynchronous video that it offers [10]. In that context it also marked some interesting advantages with respect to the in-person form.

Among the various solutions provided, *Microsoft Teams* has been one of the main instruments for companies in need of a common space of work, and not only. In fact, we have seen many institutional fields adopting Teams as platform to provide group chat, shared files and conference calls during this tough year. For our concern, where the platform has been particularly efficient was in providing support for companies during the decentralized work period. In many cases Teams allowed big and small agencies, sometimes with no IT background at all, to have an efficient instrument for different purposes. In this way, the platform became used not only for communication, but a central hub for daily user activities, giving in the end also the chance for the implementation of dedicated app, integrated with the enterprise context of the company.

For all these reasons, the goal of this thesis is to build a solution for the integration of an external functionality into one of this communication platforms, in particular into Microsoft Teams. We will exploit a real use case, the request of a client company with an intranet portal on premises that had to be integrated, to explore the potential of these tools. Our fundamental goal is to develop new instruments for everyday work following its evolution in relation to WFH, instruments that can be integrated and interact with this constantly used platforms and still providing additional functionalities, different from the communication ones. In this text we will study in deep the technology that nowadays we could adopt to develop this kind of solutions, inside chapter 2, then we will present the use case and the design between this work (3-4); after that, we will focus on the implementation of our solution (5), then the obtained result (6) and finally the conclusions and possible evolutions (7).

Chapter 2

State of Art

2.1 The impact of the pandemic on working from home

The implications of this particular situation are not yet fully understandable, and we will have a clearer view of the consequences during next years. Many effects of the pandemic on people working from home (WFH) are strictly related to the pandemic itself, but still some effects of the increased WFH can be isolated: "*increase in block of free time, which people can use to focus on their core tasks*" and "*workers could be collaboratively isolated as meeting time decreases"* [3]. One of the questions that arise from these is: can companies continue to innovate with a different equilibrium between deep thought and exchange of ideas? This situation also created the opportunity to design new productivity tools specifically for remote work, [3] going beyond communication aspect, that is our main topic in this text.

It is important to also underline the result of some experiments, that marks how *working from home actually improves productivity*. From a Chinese experiment [4] we can see a 13% performance increase in a company with 16000 employees, leading the company itself to roll out the option to work from home to the whole firm.

Beyond the actual productivity increase, during this pandemic period working from home has been the only possible choice to keep people safe in many different fields. We can see how workers in US have changed their work life with some charts [7].

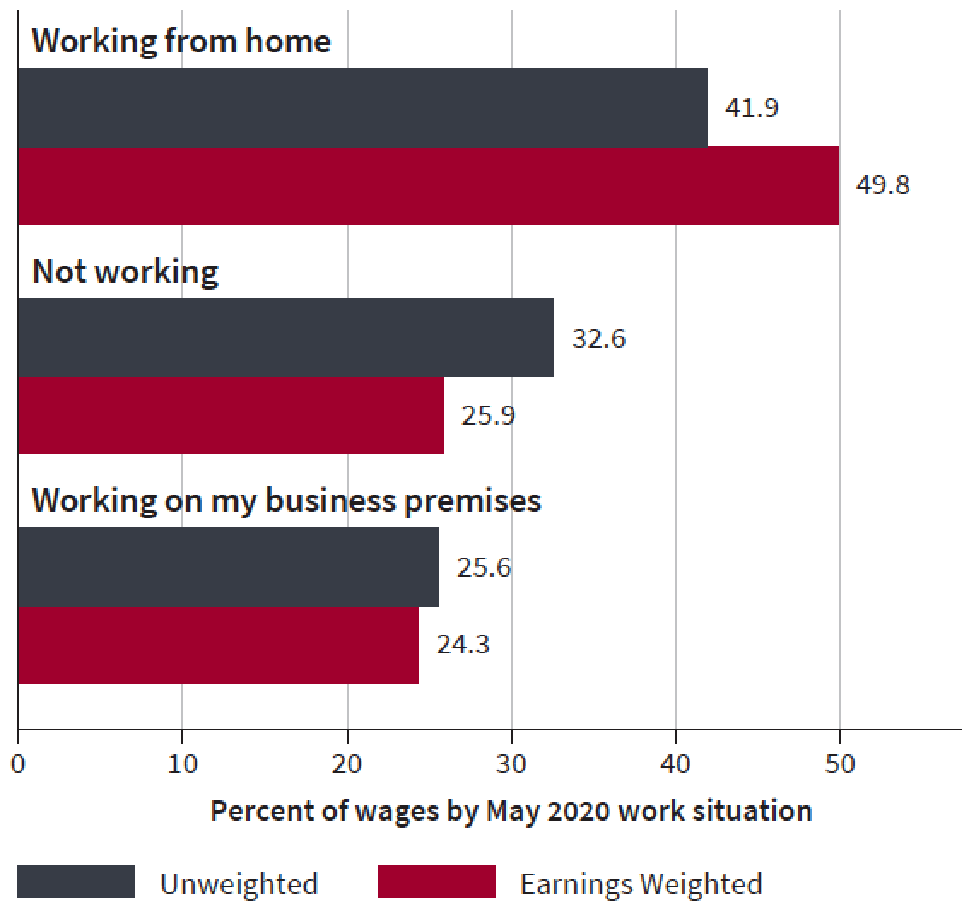


Figure 2.1 Percent of people WFH. Image taken from [7]

From figure 2.1 we can see how the majority of people (42%) is now working remotely, while almost the 33% is unemployed due to the recession. The implications of this situation are uncountable, and it is possible to analyze it from many points of view. Which works can be actually done from home? How comfortable is working from home? Which role will WFH have after the end of the pandemic? For example, it is interesting that the percentage of people working from home is higher in educated higher-income employees, as we can see from the figure 2.2 below.

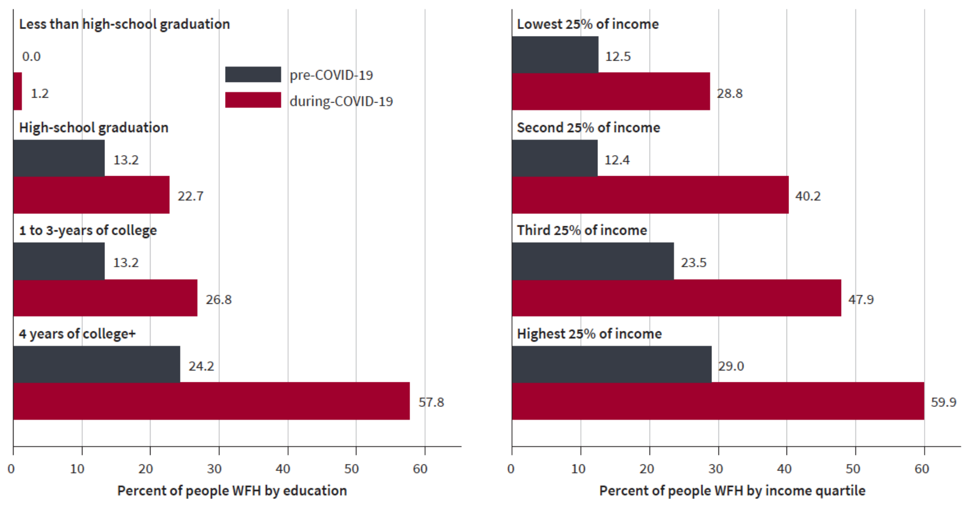


Figure 2.2 Percent of WFH employee per social class. Image taken from [7]

2.2 Working from Home after COVID

The interesting thing is that working from home is here to stay [7]. It is important to understand the impact that this can have on city centers, were much of the workers were concentrated in pre-COVID world. This could have a big depressing effect on the cities all over the world, reducing the daily expenses in bar, restaurants or shops. And obviously this money would impact instead on the suburbs and rural areas, inverting a trend going on at least since 1980s.

Moreover, WFH have already changed the perspective of work we have today, during pandemic, and will change it even more during next years. Before 2020 only a small percentage of workers have tried work from home, and it was generally stigmatized and seen negatively. Nowadays, during pandemic, not only it is a perfectly normal situation, lived everyday by a lot of people, but it brings also many new problems. For example, many employees have children at home, that brings to a lack of space, or quiet space. COVID has forced a lot of people to work from home under terrible circumstances, and that’s something that must be considered for the future of remote working.

What is really interesting is thinking to WFH in the post-COVID world, that is the new focus around this topic for future years. Before the pandemic on average only 5% of working days were spent at home, while now it is about 40%. After pandemic the number will likely drop to 20% [7]. It is still a lot higher than the pre-COVID situation, underlying that, as said, WFH is here to stay. While the number of companies that intends to keep full time remote work is small, almost every company has been positively surprised by how well it has worked [7], and has intention to maintain at least some days per week of WFH.

Although, the discussion around the advantages and disadvantages of working from home has going on during the last five years, at least. However, many points are still definitely open, like safety and health of the worker, or balance between work and non-work time. In this sense, we should know in advance the tradeoff that working from home implies, and prevent some critical related issues [5].

The discussion around the remote work is not only about whether working in remote or not. An open point of discussion is also on the differences of working from home and working from anywhere, and the possible advantages of both situations. The second solution, WFA, refers to the possibility that the employee works not only from home, but from wherever he or she wants to live. This implies that people can live in a very distant place from company's offices, that allows highly skilled workers to choose freely where they want to settle down. A particular study on the difference between WFH and WFA [6] has underlined an interesting increase in productivity in the second situation of a +4.4% in work output.

2.3 Microsoft Teams as a working from home communication platform

In this context, communication platforms have seen an incredible increase in daily users, messages and calls. The data about the growth of Microsoft Teams (figure 2.3) speak by themselves. According to Microsoft, only in the month of March 2020 the daily minutes spent by users on Teams have raise from 560 million – 12th of March – to 900 million – 16th of March – reaching at the end of the month 2.7 billion. These data alone give a clear idea of the exponential growth that the WFH periods meant for the platform.

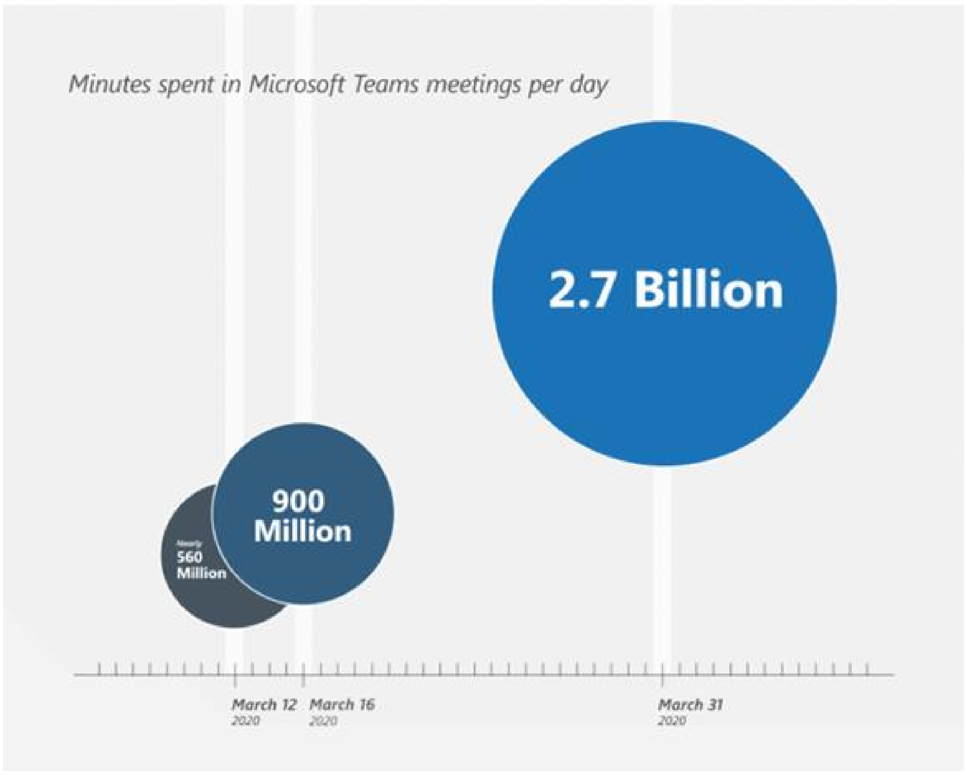


Figure 2.3 Teams' growth in March 2020

According to another official report [17], on the month of September 2020 Microsoft Teams had incremented its number of chats per user of the 48% and the weekly number of meetings and calls of the 55%. We are talking about a platform with more than 110 million daily active user - against the 32 million of march - and more than 200 million daily meeting participants. Teams is the fastest-growing business application in Microsoft history.

One of the main strength of Teams, compared to the competition, is being part of the Microsoft 365 platform. This service by subscription of the company from Redmond, until recently known as Office 365, contains all the main software that made Windows the most famous operative system in the world, and it integrates those with many other services and functionalities. Thanks to this Software as a Service, Microsoft gives to its subscriber all the potential of its vast suite. In this way we can integrate the Teams platform with many others, like Outlook for mail managing, SharePoint for file sharing, and all the classical software of the office packet, from Word to PowerPoint, from Excel to OneNote. All of these with the possibility to co-edit in real time the shared files.

Within Microsoft 365, Teams was originally designed to inherit the functionalities of Skype for business, with its main focus being communication. Afterwards, thanks to some key functionalities, the platform evolved to become a central hub to everyday work life. Teams became an instrument not only for communication, but basically a space of coworking for employees' activities related to work, and also a meeting point for colleagues where to share everyday life.

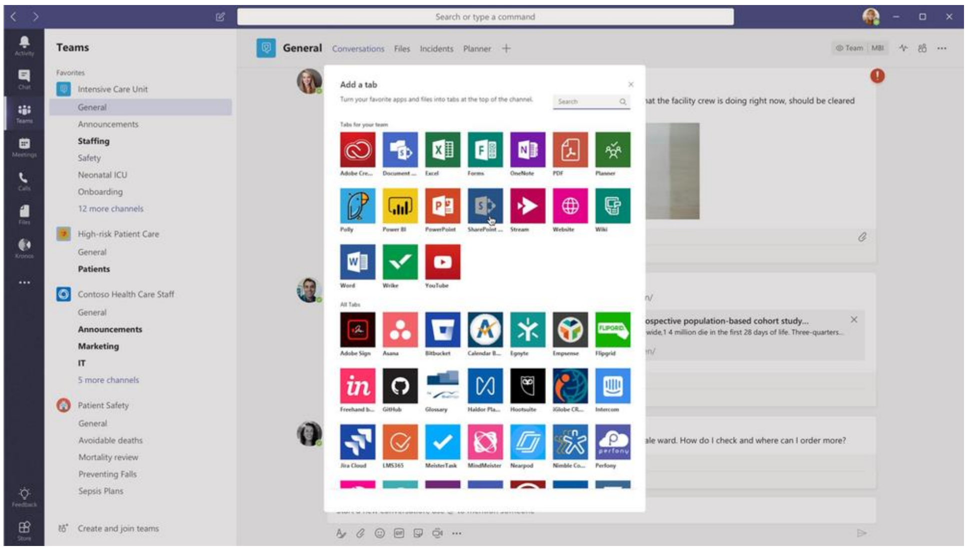


Figure 2.4 Microsoft Teams app store

The platform allows a variety of operations designed to facilitate the work life of the employees. In fact, it becomes a space of work thanks to the possibility of creating teams and channel, where not only chat with colleagues, but also upload files, automate task using bots, take advantage of the notification systems. Teams allows to create shared documents, giving the chance to edit them in parallel, using the software of the Office 365 suite, it allows to plan the calendar and schedule and monitor teams tasks, as well as team members holidays, we can integrate timesheet functionalities, desk/room booking functionalities, ticketing and many more. The possibilities are uncountable, taking into account also the chance to extend the platform exploiting the thousands of Microsoft and third-party applications that we can integrate. Moreover, Teams offers social-like functionalities, like profiles, likes, replies, GIFs, sticker, emoji.

2.4 Teams integration with other Microsoft instruments

A great advantage of Teams as a communication platform is its being part of the Microsoft environment. This opens up to many possible combination with other application and services offered by this environment, in many field. We can exploit the powerful tools provided for cloud technology, such as integrate with the classic application for everyday work life.

One of these fundamental instruments, part of Microsoft 365 that is fundamental in this technological context is the Graph API. Microsoft Graph is a RESTful web API that allows our application to access the resources of the cloud. Its use enables us to register the application and exploit an authentication system based on tokens generated for a user or service as consequence of a request to the Microsoft Graph API. This functionality uses the HTTP method of the requests to determine what each one is doing. It supports most of the classic HTTP methods, such as GET, POST, PUT or DELETE, in some cases allowing the use of a body for the request, usually specified in JSON format, that contains additional information on the request.

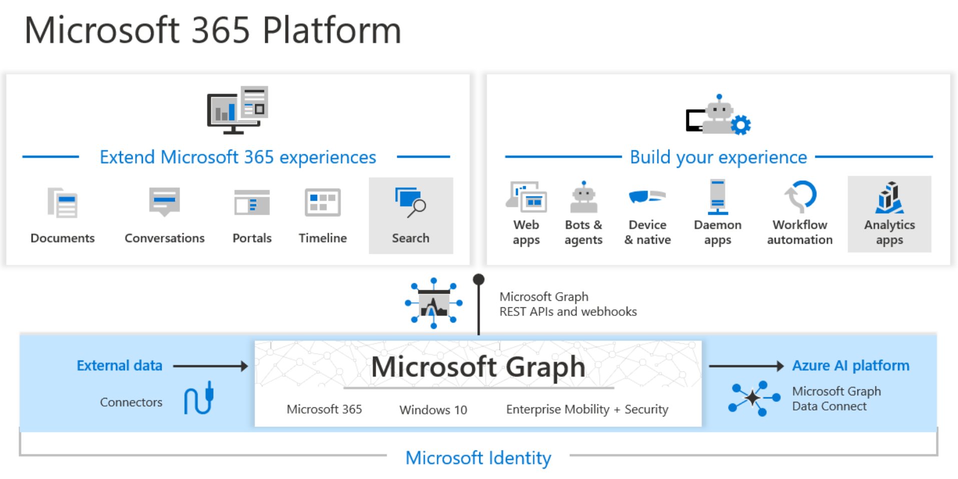


Figure 2.5 Microsoft Graph as interface between Microsoft 365 and the Cloud

Actually, Teams also allows an easier integration with all the technologies offered by Microsoft Azure. For example, we can use the Azure Logic app, a platform that exploits the cloud to provide automated workflows that can integrates many services and systems. The use of this technology allows to easily integrate a scalable solution to connect a local legacy system with the environment of Microsoft Teams. But not only: using this technology we could integrate action related to any of the applications of Office 365, building an ad-hoc trigger that will activate after a specified event; finally, we could also use the workflow to create a monitorization system, to analyze sentiment on social networks or other similar platforms.

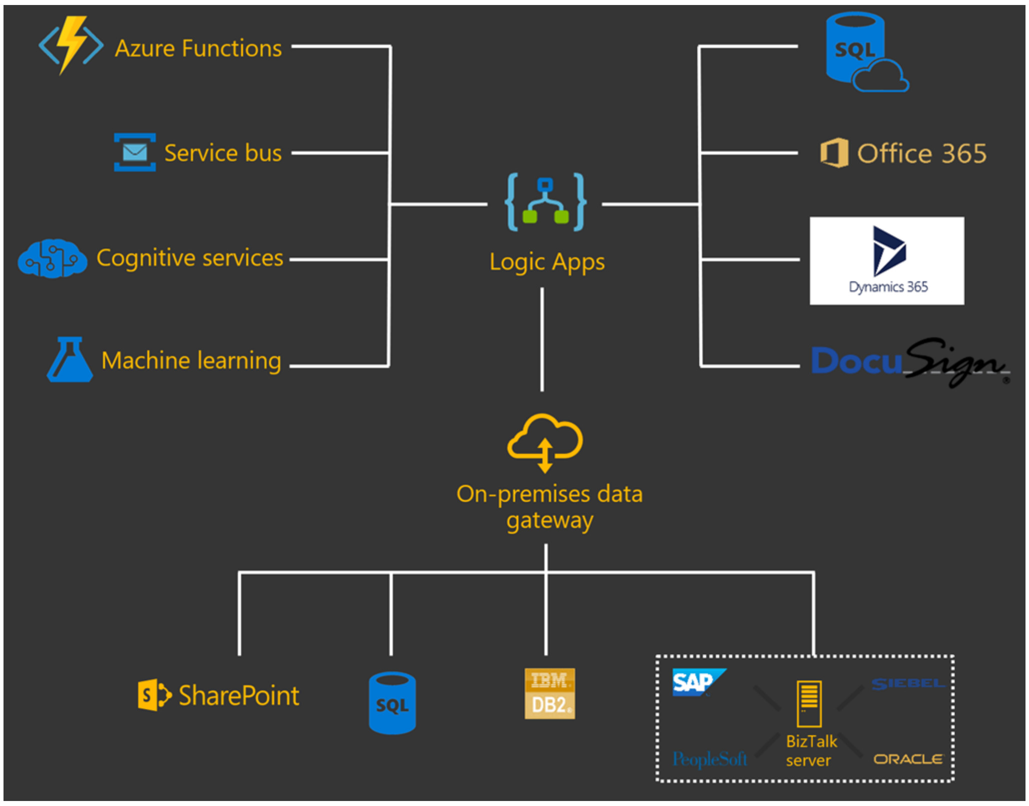


Figure 2.6 Logic Apps as a connection tool

In this context, another interesting technology that could ease our work is the Adaptive card. These cards are an open format for the exchange of information with graphical user interfaces of many different types, using a unique and coherent way. We can use them exploiting JSON object to build contents that will be rendered differently depending on the host application to which we will send the card. Microsoft Teams will render them coherently with its own graphical aspect, familiar to the final user.

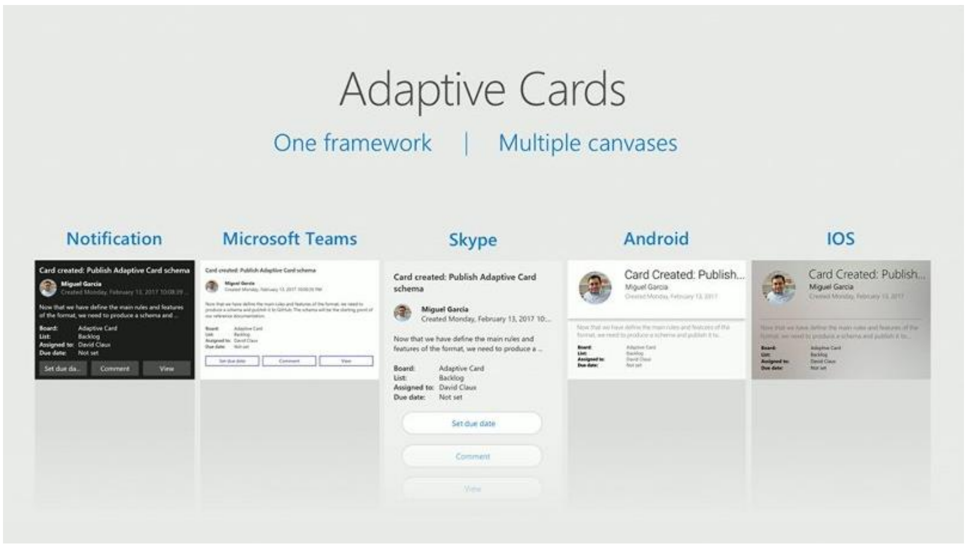


Figure 2.7 The versatility of Adaptive Cards. Image taken from [13]

For these reasons, in many situation companies felt the need to integrate inside an all-in-one platform, as Teams, all the tools related to business and cooperation.

2.5 The importance of the Cloud

While we already explored the different platforms dedicated to remote and immediate communication, we did not talk about the cloud platform dedicated to this solution. Nowadays, the cloud technology is widely used in many different fields and in many ways. There are many providers of cloud services, and three specific models in which we can classify these services: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). The differences between these three kinds of services are mainly the degree of responsibilities that the final user has. For example, in IaaS the service provider offers an empty platform, like a VM, and the user has to manage it, installing the operating system and the software needed. On the other hand, in the SaaS model the provider can offer to the user a specific software, without any customization, used on demand.

We need to develop an external tool, an app that has to be integrated in a communication platform. So, for our purpose the Software as a Service model is the perfect choice. We can deploy on a cloud service hosting a custom application that we will then integrate in the chosen team speaking software. Alternatively, we can use the instruments offered by Cloud to create an automated workflow to connect our existing portal directly to Microsoft Teams

The platforms that offer the SaaS hosting are many and different. Among the most used we have Amazon Web Services, Google Cloud and Microsoft Azure. Each of these platforms can be used for our needs.

These cloud platforms offer also a lot of interesting services for every needs. For example, Microsoft Azure offers Azure Active Directory, a platform created to manage and protect identities and all the process related to it. With it is possible to provide many functionalities to companies, like the Single Sign-On and the multi factor authentication. With AAD we can concentrate in a single platform all the identities related to our solution, both internal and externals, and most importantly for our concern we can integrate in a simple way app and services using the same identities.

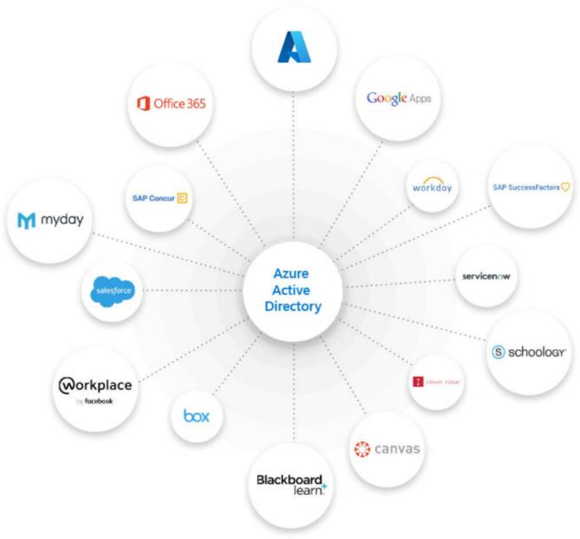


Figure 2.8 Azure Active Directory as authentication service

Starting from this technological context, we thought about something to exploit the potential of the videoconferencing and communication platforms as an instrument to improve and ease everyday work life. The central idea was to enhance the capacity of one of these platforms to become a central hub for every activity related to work, and because of this we wanted to experiment with the possibilities of integrating external services inside this hub. In our idea of remote working, this could draw a line in the evolution of the daily routine of the employee working from anywhere.

Chapter 3

Use case

3.1 The starting point

It is in this context that a real use case presented to us. A company with more than 65 thousand employees, with an intranet portal dedicated to internal communication, needed to integrate its functionalities inside a fresh, portable and more interactive system.

The existing portal was used to communicate with the employee daily information about the whole company or specific divisions, and at the same time to give an instrument to keep updated companies' departments on the latest news. For its purposes, the portal used emails as notification mechanism, even though it had social-like integrated functionalities, such as 'like' and comments.

The existing intranet portal is built around a web-based application and the concept of news. Through the news the company can manage the internal communication even though the internal teams are scattered all around the world. Inside this intranet the news could be "global", such as the success of an important company's project, or "local", such as communication on how to behave after a geographically significative event. These news are based on the scope of interest and relevance for the end user audience. So, the whole web portal is based on a system of audience that allows to show the news to the specified group of users. In the "as is" situation the news are just a window on some information that the company needs to share with its employees.

Global news is managed by an internal communication "global" team, that has the power to publish also local scoped news or delegate the publication to a local team. Local teams take care of internal communication for a specific geographic area or production plant.

The big limit of this configuration is that the Intranet portal is, by definition, only accessible from within the borders of the company network. It is not possible to access the intranet from home because it is not "published" outside. It must be accessed from a private network, physical or virtual. Another limit of this intranet, that with our solution we want to overcome, is the missing support to mobile devices.

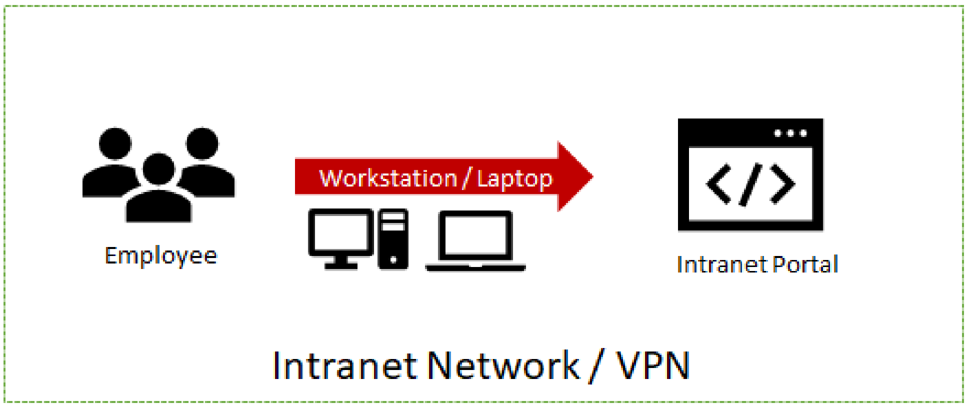


Figure 3.1 Access to the news in as is situation

3.2 The innovation steps

To overcome the limitation the company decided to investigate possible innovation that allow to support modern scenarios, that along with working from home are no longer a will but a productivity need. The idea was to allow the access to the intranet portal from home, exploiting the new platform offered by Microsoft in terms of both security and communication. At the same time the access from mobile devices was a desired feature.

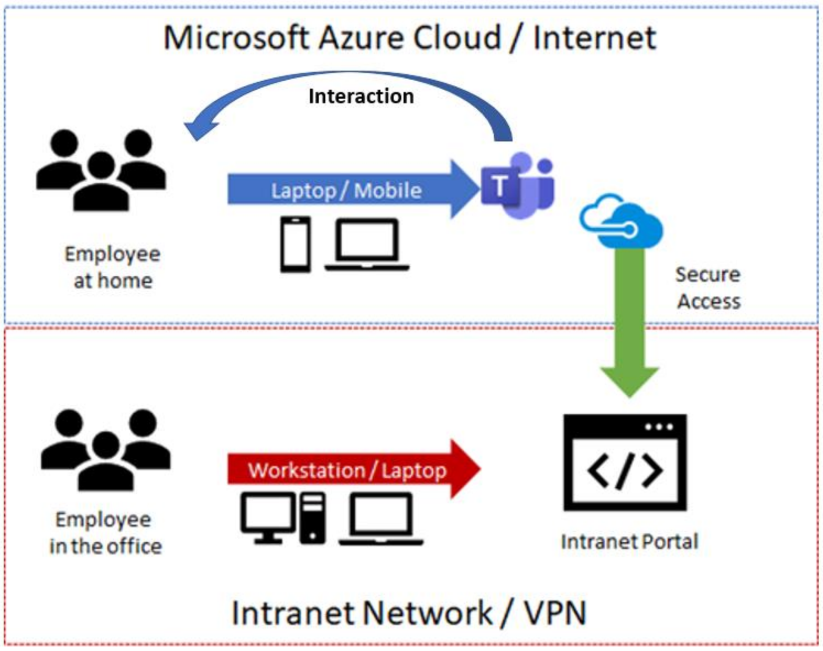


Figure 3.2 Access to the news in to be situation

In this context, both global and local team are constantly looking for new ways to interact with the employees, trying to obtain a more informal and immediate communication stream. The company was already using Teams as communication platform, and it represented one of the new channels on which the communication of the company is focusing. As a natural and innovative extension of the “legacy” web portal approach, MS Teams “chat like” system allow a more informal and streamlined communication channel and also a change of paradigm of communication. Using MS Teams, it is possible to push relevant news directly to the right audience; the end user no longer needs to visit the web portal to reach the news feed, they just will change the tab of their already opened Teams window.

Moreover, Cluster Reply’s Microsoft partner and develops in a Microsoft environment. For these reasons, we choose to integrate the news mechanism into Microsoft Teams, not to totally replace the existing portal, but to provide an alternative to the company and its users. In this way we could leave the choice to the final user about how to exploit this information. He could continue using the existing portal as is, without changing his habits, or alternatively he can try the new solution, that gives him the chance to consult the news in a quicker and easier way.

3.3 “As is” and “to be” situation

The “as is” situation of the intranet portal provides a web interface that allows editor to create, publish and manage news, giving the possibility to choose whether a news must go on the homepage or not. From the news publication screen, after choosing a title, the editor is able to edit the details, such as the subtitle, the text, the image and other information regarding who can see this news, depending on the zones or on the teams.

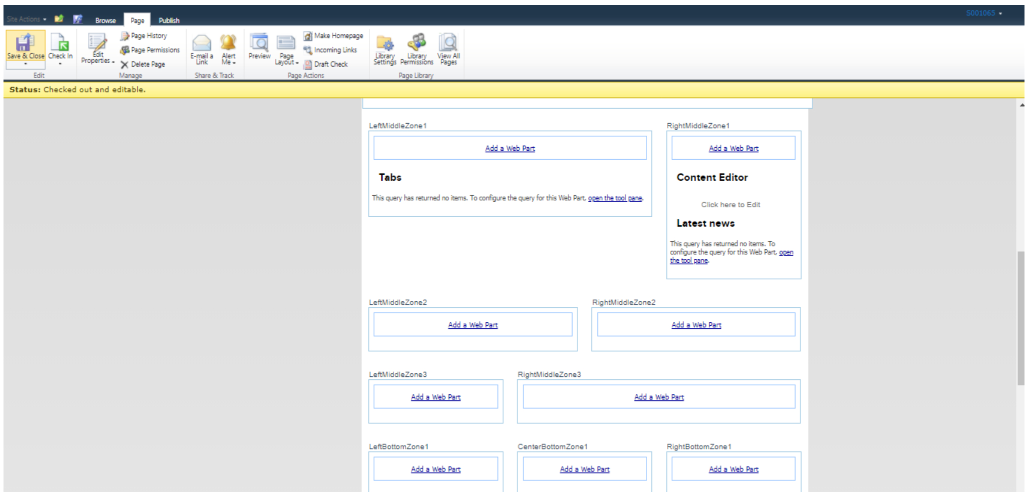


Figure 3.3 "As is" situation seen by editors that must create a news

From the final user point of view, he will have access to a personalized screen of the portal, showing the news that he's allowed to see, depending on his membership to an audience. This audience, as said, will depend on the region where the employee is set and on the teams he is member of.

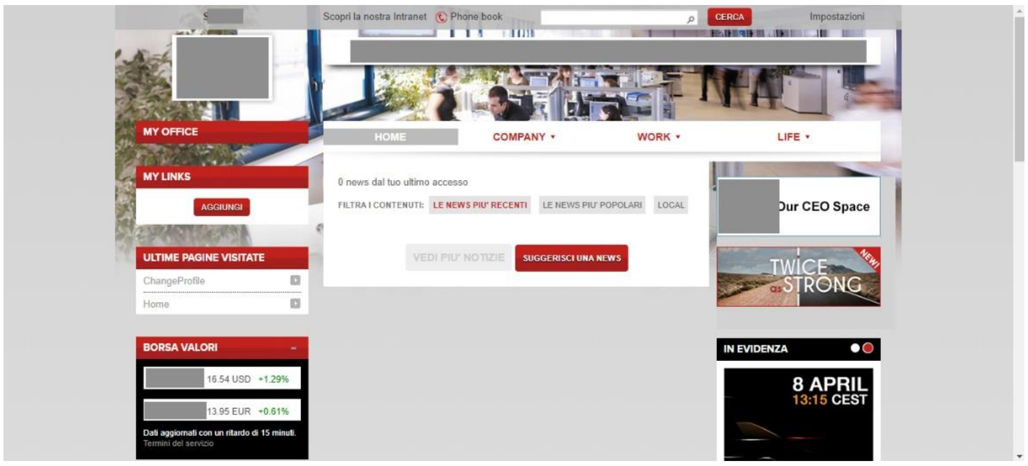


Figure 3.4 "As is" situation seen by the final user

Chapter 4

Project Design

4.1 “To be” goals

The goal of this thesis is to explore the potential of video communication platforms and the benefits they could involve if adopted in a business context. The question is: how can they ease and promote the work from home? In our case we thought to a situation where a company was already using some software dedicated to communication and wanted to integrate it with some external functionalities.

As said, the company had an existing intranet portal. It was on premises and reserved for its employees, and its management was demanded to Cluster Reply’s. This allowed us to modify it, creating dedicated functionalities that could provide information to an external platform, and so we could think to an alternative solution to the existing one, a solution that could be more immediate and comfortable. In this sense we could create a hook mechanism directly on the portal, with a dedicated endpoint, enriching the existing solution with code specific for our needs.

Regarding the “to be” system, the goal was to provide an alternative and more immediate fruition of the news information, without complicating the workflow of the publication. So, what we wanted to achieve was an interface totally integrated with Microsoft Teams, that could allow to transfer the exact same news published on the portal also on a dedicate tab inside the communication system. All of this without changing the process of creating and publishing the news.

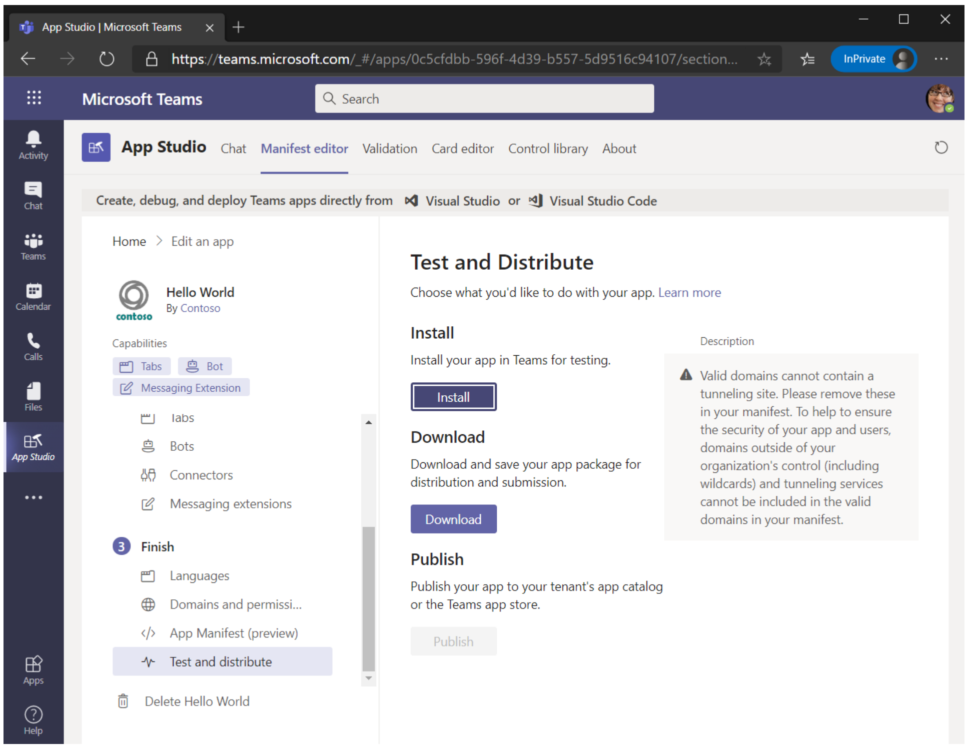
4.2 Two alternative solutions

The access to the news portal in the as is situation needed to be via web browser, while the new philosophy behind the to be solution is to move everything into an already work-focused environment. Teams is an always open platform for the employees of the company, an instrument that constantly ease their work lives, providing them with all the functions that they need. The idea behind this project is to adapt to the change of mentality of the company and of the whole world of work providing a concentrated, easier and quicker environment, that can increase at the same time the productivity and the quality of life of the employees.

In this thesis we propose two different solutions to solve the problem of integrating and improving the news infrastructure. The first one is based on a dedicated Teams application, that could interact with the already existing services on the old system to get the news and show them exactly as they were shown in the original portal.

4.2 App Integration in Microsoft Teams

Microsoft Teams apps are an instrument that gives developers the chance to integrate an existing web app into Teams, or to realize a custom service to access common tools or specific information and integrate it in the communication platform. Realizing an application for Teams we could customize the experience of the employees in many ways. The goal was to make their lives easier, concentrating everything they need to work in a single place. We can exploit the possibility to use tabs to highlight some important content that users will often need handy. These custom tabs are basically webpages embedded in Teams, where the domains where to point are declared in the manifest of the app. Here, in the manifest, as developers we can choose if the app will be used as a channel inside a team, in a group chat or as a personal app. The platform gives many tools to develop third party apps or integrate an existing one, allowing to embed a new or already existing custom functionality to our teams or personal employees' chats.



Create, debug, and deploy Teams apps directly from Visual Studio or Visual Studio Code

Test and Distribute

Choose what you'd like to do with your app. Learn more

Install

Install your app in Teams for testing.

Download

Download and save your app package for distribution and submission.

Publish

Publish your app to your tenant's app catalog or the Teams app store.

Valid domains cannot contain a tunneling site. Please remove these in your manifest. To help to ensure the security of your app and users, domains outside of your organization's control (including wildcards) and tunneling services cannot be included in the valid domains in your manifest.

Figure 4.1 Microsoft Teams' App Studio

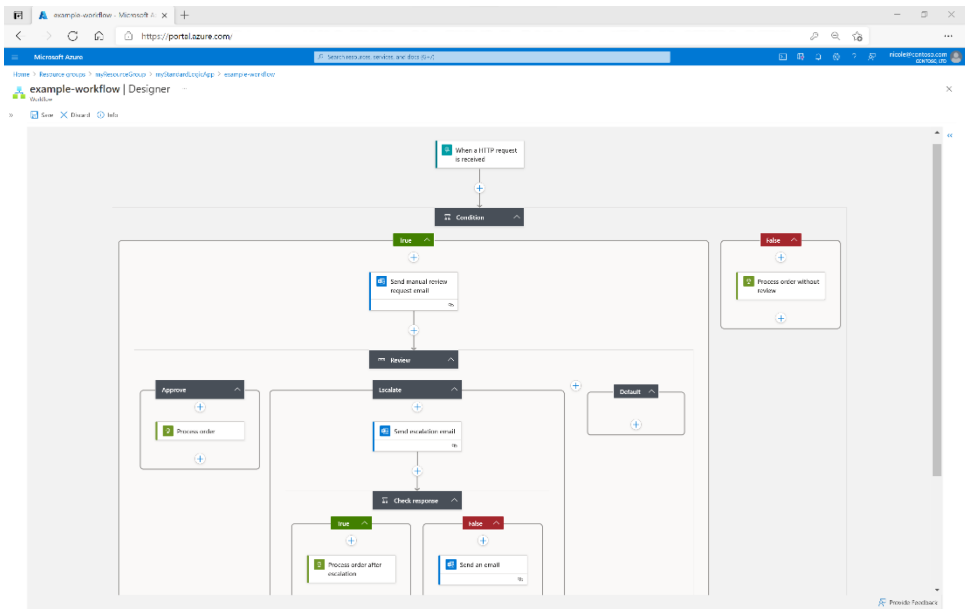
Our work for this first solution is focused on the design of a dedicated web application that could replicate the interface of the existing intranet portal inside a tab on Microsoft Teams. To obtain this result, we had to realize an authentication system that could connect the users directly with their MS credentials to the web app, and that could also use those credentials to automatically log him on the intranet portal in the background to retrieve his personal news. This crossed authentication was realize using Microsoft Graph API, that could allow us to reach our goal and automatically authenticate the user to the external web app with his Teams credentials, connecting the app with his teams, channels and others information related to the communication platform. To obtain the cross connection we needed Graph, as said, and to use it in combination with our Teams App we needed Azure Active Directory.

From the intranet portal point of view, with this solution we would have to create an ad hoc endpoint on the news service to manage this external access to information. Since the news are stored in the database of the existing web portal and all the infrastructure of it will remain unchanged, the best solution to obtain the data is to create specific methods to call, or to modify the existing one. In this sense we thought about creating dedicated functions to retrieve the data for the Teams App, since we need to manage calls coming from outside the portal, without an already authenticated user defined in the context. In this way we could have send the credentials of the user from the app inside a header of the REST call and manage the authentication of the user in an invisible way inside the specific method.

4.3 The second alternative

This other solution is based on a different approach to integration. Instead of integrating an application that manages the interaction with the user and with intranet portal, here the goal is to simply push the news automatically as post in a dedicated channel. In this way, all the logic remains in the portal and we need less changes to the code. Still we need to create an external mechanism that triggers on publication and react recreating the news inside Teams.

Taking advantage of the already mentioned Logic Apps we could realize an automated and scalable workflow that could be triggered by a specific action. Logic Apps are a cloud service from Azure, an instrument to automate company processes and workloads. They expose a range of API as Connectors in order to define these automated workflows. With them we can plan and send notification using emails through Office 365 as a consequence of a specific event; we can elaborate customer’s orders and send them on local or cloud services; we can move files on SFTP or FTP server to Azure storage; we can monitor tweets and other kinds of post on social networks, in order to analyze the sentiment and give advice and activities for elements that need to be monitored.



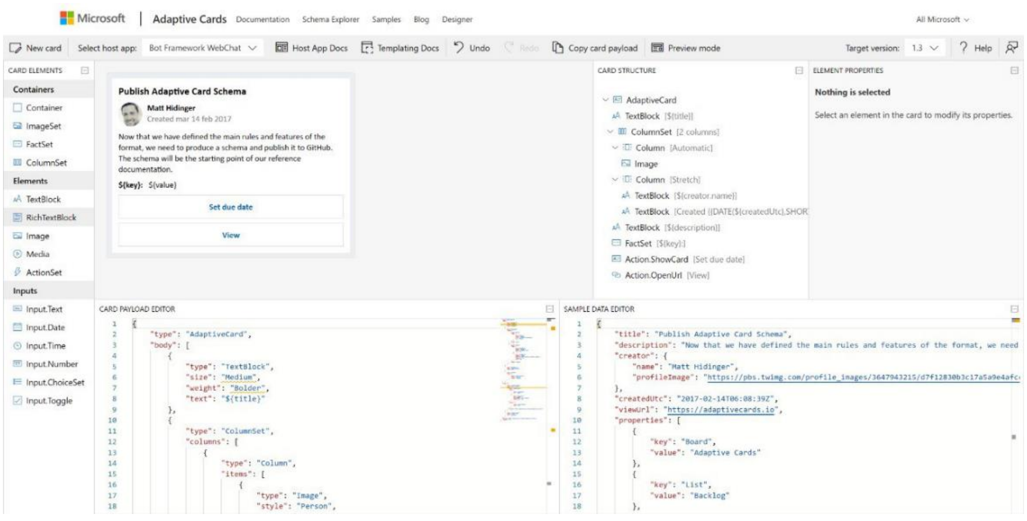
https://portal.azure.com/
Microsoft Azure
example-workflow | Designer
Save
Discard
Info
When a HTTP request is received
Condition
True
False
Send manual review request email
Process order without review
Review
Approve
Process order
Localize
Send escalation email
Default
Check response
True
False
Process order after escalation
Send an email
Provide feedback

Figure 4.2 The Logic App designer in Microsoft Azure

The instruments offered by these Logic Apps are basically of three types: Triggers, Connectors and Actions. The first ones are the starting point for the workflow of an Azure Logic App; they will fire when new data or a specific event that meets the trigger specification occur. We can create custom trigger or simply exploit the basic one offered by the service, and we can also use multiple triggers to activate our app. A Connector is used to create the real flow to perform actions or processes and connect users' data to the advanced operations of the flow. Actions are instead the actual operations that must be performed by the App after it gets triggered. They can compute and elaborate data for the next Actions, in a sort of chain that finally will generate the expected result. Speaking of Logic Apps, we should not forget to mention also the possibility to integrate them with many Microsoft technologies, in single or multi-tenant environment, that makes them extremely versatile and powerful.

About the available actions that a Logic App can perform there is also the possibility to send some Adaptive Cards to different external platforms, as for example our beloved Microsoft Teams. This technology was released by Microsoft in 2017 and is basically a universal Card user interface framework. Its objective is to provide a standard way of defining cards, independently from the platform. In this way we could build the body of the card just once and then use it on different platforms: it will automatically adapt to the graphical interface on which it will be sent, if this supports the Adaptive Card. These cards are written in JSON and they are basically a recipe on how to structure the card, that will be interpreted by the platform where it will be sent. So, the same card could have a different look depending on where it will be showed, while providing the exact same functionality.

These cards offer a lot of functionalities, not only about showing information but also about retrieving it and allowing the user to interact with the server. Moreover, they allow to simply embed in the card the data from the JSON that will define the card. Adaptive Cards are a very powerful instrument that also offers a very simple user interface to design them. In our case, we could use the dedicated website to build the perfect card for our needs and then simply copy and paste it on our Logic App workflow, so that the action will send the cards to Microsoft Teams.



Publish Adaptive Card Schema

Matt Hidinger
Created mar 14 feb 2017

Now that we have defined the main rules and features of the format, we need to produce a schema and publish it to GitHub. The schema will be the starting point of our reference documentation.

${key}: ${value}

Set due date

View

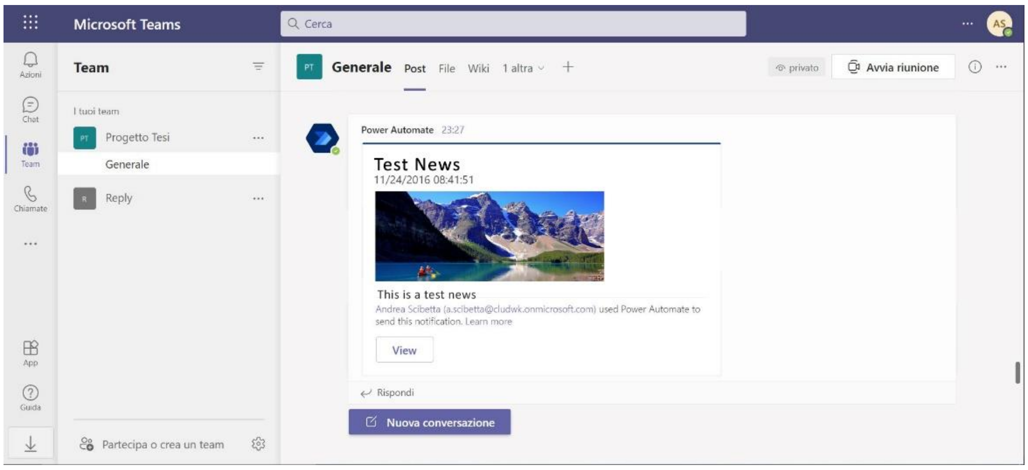
```
AdaptiveCard
  TextBlock [${title}]
  ColumnSet [2 columns]
    Column [Automatic]
      Image
    Column [Stretch]
      TextBlock [${creator.name}]
      TextBlock [Created {{DATE(${createdUtc},SHOR
  TextBlock [${description}]
  FactSet [${key}:]
  Action.ShowCard [Set due date]
  Action.OpenUrl [View]
```

```
{
    "type": "AdaptiveCard",
    "body": [
        {
            "type": "TextBlock",
            "size": "Medium",
            "weight": "Bolder",
            "text": "${title}"
        },
        {
            "type": "ColumnSet",
            "columns": [
                {
                    "type": "Column",
                    "items": [
                        {
                            "type": "Image",
                            "style": "Person",
```

```
{
    "title": "Publish Adaptive Card Schema",
    "description": "Now that we have defined the main rules and features of the format, we need
    "creator": {
        "name": "Matt Hidinger",
        "profileImage": "https://pbs.twimg.com/profile_images/3647943215/d7f12830b3c17a5a9e4afc
    },
    "createdUtc": "2017-02-14T06:08:39Z",
    "viewUrl": "https://adaptivecards.io",
    "properties": [
        {
            "key": "Board",
            "value": "Adaptive Cards"
        },
        {
            "key": "List",
            "value": "Backlog"
        },
```

Figure 4.3 The Adaptive Cards designer

Exploiting the logic app and the adaptive cards we can obtain this automated solution, that should in the end show an interface as the one in figure 4.4. This tab inside the selected channel will be filled with the news after each new one is published.



Power Automate 23:27

Test News
11/24/2016 08:41:51

This is a test news

Andrea Scibetta (a.scibetta@cludwk.onmicrosoft.com) used Power Automate to send this notification. Learn more

View

Figure 4.4 "To be" situation seen by the final user

If we want to replicate the system of audience, we should prepare a set of different channels, publishing the news through our logic app based on which users should have access to them. Also, we should give access to these channels respecting the user who should see the news contained inside it. While this solution need less changes to the intranet portal with respect to the first one, it will need an extra effort to recreate this kind of “profiling” solution on the external platform.

After the researches that brought us to design these two solutions and define the technologies needed to develop them, we started working on the real implementation of them.

Chapter 5

Implementation

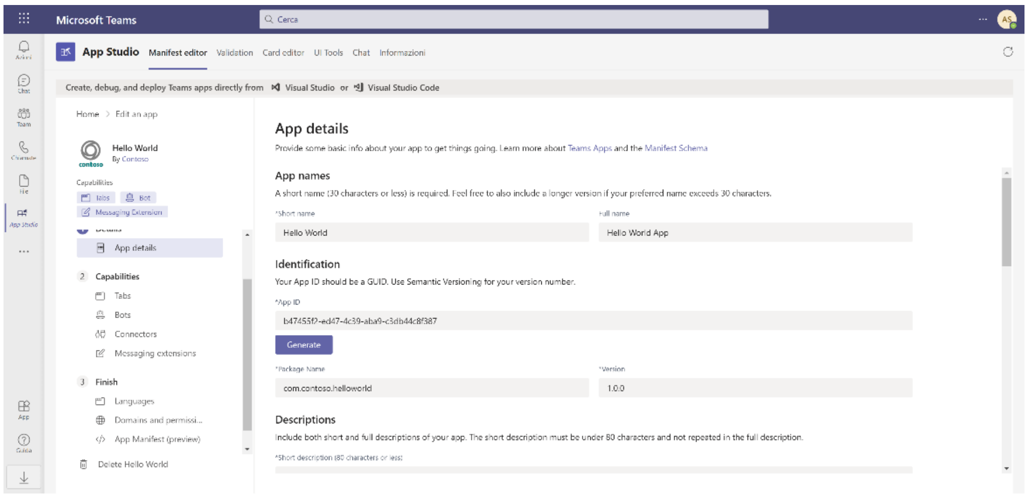
At this point we had a clear overview of the technologies that could help us to fulfill our goal, and some ideas on how to build a solution. At the beginning we started working on the idea of building a Teams App, that as we said is basically a web application integrated in Microsoft Teams through its App studio.

5.1 Web Application development

Starting from a simple application, built following the proper guide [15], we studied how to enable the mechanism of integration at its core, and how to create an authentication system connecting our Teams app with the intranet portal through Microsoft Graph API.

We deployed the application to Azure, to have a unique address to reach it and connect it to other systems. After that, we could follow the procedure to define the Teams App Studio association, so that our web application could be installed into different teams or channels. Following the steps, we had to save some specific data from the Manifest Editor that will be needed to enable the authentication with Teams Credentials.

At this point we define all the information related to our App, from the necessary and fundamental details, such as the name, icon or ID, to additional one, such as tabs definition, with their route, or bots. These last one, such as Connectors and Messaging Extensions, are instruments that we will not use in this project, but could be very useful to implement other functionalities in the future and improve the existing one. Using them we could also provide a better integration with the platform.



Create, debug, and deploy Teams apps directly from Visual Studio or Visual Studio Code

App details

Provide some basic info about your app to get things going. Learn more about Teams Apps and the Manifest Schema

App names

A short name (30 characters or less) is required. Feel free to also include a longer version if your preferred name exceeds 30 characters.

*Short name: Hello World
Full name: Hello World App

Identification

Your App ID should be a GUID. Use Semantic Versioning for your version number.

*App ID: b47455f2-ed47-4c39-aba9-c3db44c8f387

*Package Name: com.contoso.helloworld
*Version: 1.0.0

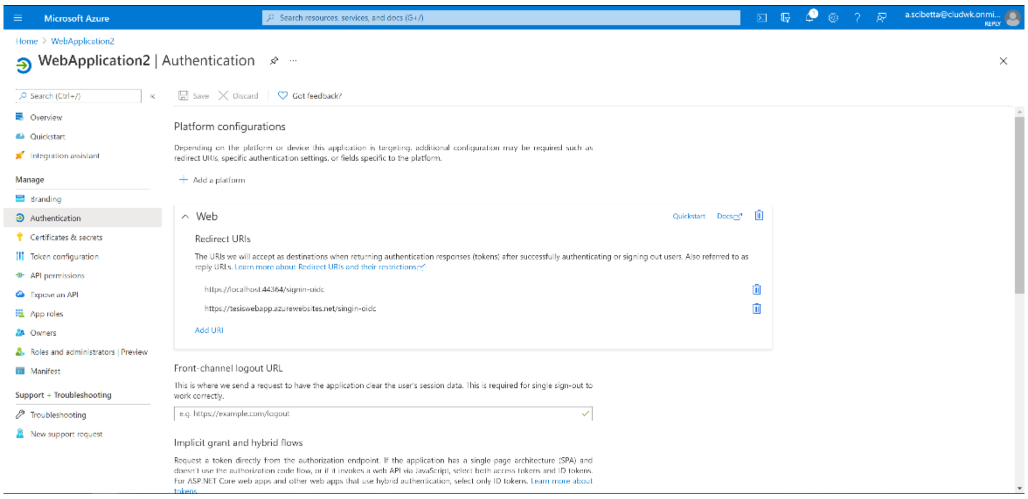
Descriptions

Include both short and full descriptions of your app. The short description must be under 80 characters and not repeated in the full description.

*Short description (80 characters or less)

Figure 5.1 Definition of App details in Teams App Studio

To enable this crossed authentication we must go through the Microsoft Graph API, as explained previously. To do this, we also need to create an App Registration into Azure Active Directory. This component of Microsoft's Cloud gives to developers the Microsoft Identity Platform, a service of authentication that offers many open-source libraries and others instruments to manage applications. Combining these tools, we decided to create an App Registration in Azure Active Directory, and use it to connect our deployed App Service with our Teams application, automatically authenticating the user on our application in the background using their Teams credentials.



Platform configurations

Depending on the platform or device this application is targeting, additional configuration may be required such as redirect URIs, specific authentication settings, or fields specific to the platform.

Redirect URIs

The URIs we will accept as destinations when returning authentication responses (tokens) after successfully authenticating or signing out users. Also referred to as reply URLs. Learn more about Redirect URIs and their restrictions

https://localhost:44364/signin-oidc
https://tesiswebapp.azurewebsites.net/singin-oidc

Front-channel logout URL

This is where we send a request to have the application clear the user's session data. This is required for single sign-out to work correctly.

e.g. https://example.com/logout

Implicit grant and hybrid flows

Request a token directly from the authorization endpoint. If the application has a single page architecture (SPA) and doesn't use the authorization code flow, or if it invokes a web API via JavaScript, select both access tokens and ID tokens. For ASP.NET Core web apps and other web apps that use hybrid authentication, select only ID tokens. Learn more about tokens.

Figure 5.2 Azure Active Directory App Registration

5.2 Connection with the existing infrastructure

We have created the structure for our Teams integrated application: we have a web app deployed to Azure with its own reachable address, linked to Microsoft Teams App Studio as an installable add on and with authentication permission on Microsoft Graph API through Azure Active Directory platform. What is still missing is the connection with the existing portal. We need a way to define a hook on the intranet portal that can recognize the user and allow him to obtain the basics information, such as the news and the related social interactions.

To build our custom code, we obviously had to rely on the existing software context of the intranet portal. The core functionality that we needed to recreate was the method that allow to retrieve the news. Since we were starting from the web portal home page, we tried to replicate the behavior of it on our Teams application. So, we took as a base for our developments the existing Get News method, that was retrieving that information to the pages of web portal.

```
static object lockObj = new object();
        public GetNewsResponse GetNews(string Filter, SortDirection Sort, int Page, int MaxItem)
        {
            try
            {
                GetNewsResponse resp = new GetNewsResponse();

                bool cacheEnabled = ListHelper.GetConfig<bool>(ConfigCategory.Cache, ConfigKey.News, false);
                string cacheKey = string.Empty;
                if (cacheEnabled)
                {
                    CacheB2EHttpApplication c = new CacheB2EHttpApplication();
                    cacheKey = c.GetVaryByCustomString(System.Web.HttpContext.Current.ApplicationInstance,
HttpContext.Current, "Audience;Lang");
                    cacheKey = string.Format("news_{0}_{1}_{2}_{3}_{4}",
NavigationHelper.GetNavigationCacheKey(cacheKey), Page, Sort, MaxItem, Filter);
                    resp = HttpContext.Current.Cache.Get(cacheKey) as GetNewsResponse;
                }

                if (resp == null || !cacheEnabled)
                {
                    lock (lockObj)
                    {
                        if (resp == null || !cacheEnabled)
                        {
                            resp = new GetNewsResponse();

                            int NewsTotalNumber;
                            List<News> newsList = EntityManager.GetNewsHomePage(Page, Sort, Filter, out
NewsTotalNumber);

                            resp.TotalNewsNumber = NewsTotalNumber;
                            resp.HtmlResponse = BuildNewsHtml(newsList);

                            if (cacheEnabled)
                            {
                                HttpContext.Current.Cache.Add(cacheKey, resp, null,
DateTime.Now.AddMinutes(ListHelper.GetConfig<int>(ConfigCategory.Cache, ConfigKey.NewsDuration, 10)),
System.Web.Caching.Cache.NoSlidingExpiration, System.Web.Caching.CacheItemPriority.Default, null);
                            }
                        }
                    }
                }
                return resp;
            }
            catch (Exception ex)
            {
                LogHelper.Log(ex, LogCategory.B2E, LogSeverity.Error, ex.Message);
                throw;
            }
        }
```

From the code we can see how this method worked. After a check on the cache, it was defining a structure for the response and then demanding the search of the news to the GetNewsHomePage method of the Entity Manager, passing to it also the parameter on how to filter the result. On this method we faced our first problem in connecting directly Teams to the existing portal. In fact, as we can see in the code below, the Entity Manager GerNewsHomePage function is taking the user id of the current user from the context. This is obviously a problem, since we want to access this service endpoint directly, and not going through the definition of a context, such as it happens after opening the website homepage.

```
         /// <summary>
        /// get news filtered for homepage
        /// </summary>
        /// <returns></returns>
        public static List<News> GetNewsHomePage(int Page, SortDirection SortDir,
String Filter, out int NewsTotalNumber)
        {
            //get max numeber of allowed news in home (1 page)
            int NewsPerPage =
ListHelper.GetConfig<int>(ConfigCategory.NewsContentViewer, ConfigKey.MaxNews, 5);

            //get 1 page of not pinned news
            List<News> newsList = null;
            if (Filter == "Local")
                newsList = GetNews(DateTime.MinValue, DateTime.MaxValue, Page,
NewsPerPage, false, false, null, null, true, SortDir,
EntityManager.CurrentUser.Uid, out NewsTotalNumber);
            else
            {
                newsList = GetNews(DateTime.MinValue, DateTime.MaxValue, Page,
NewsPerPage, false, false, null, null, null, SortDir,
EntityManager.CurrentUser.Uid, out NewsTotalNumber);
                //merge with pinned
                newsList = MergePinnedNewsForHome(newsList, Page, NewsPerPage);
            }
            LogHelper.Log(string.Format("GetNewsHomePage: News from query {0}",
newsList.Count()));

            return newsList;
        }
```

We needed to find a workaround for this obstacle, and so we created an alternative chain of calls on the web portal, that can retrieve from a single endpoint all the information needed about the news. We started from the existing GetNews and we developed our GetNewsFromTeams.

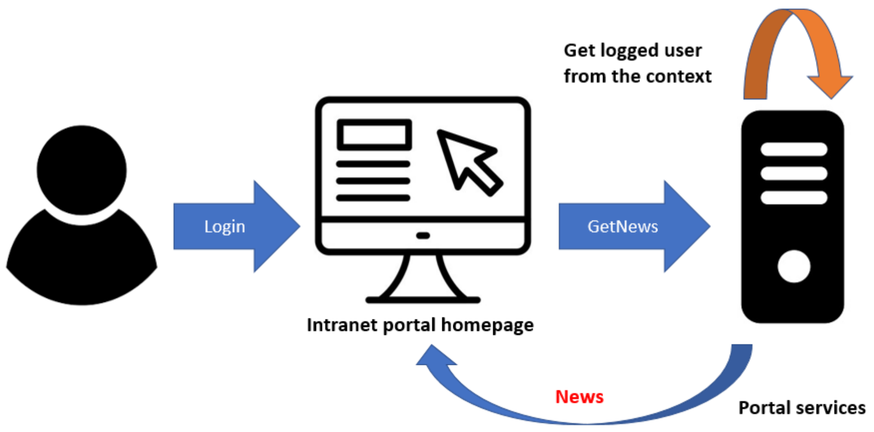


Figure 5.3 Retrieving the news in the Intranet Portal

5.2 Representing data specifically for Teams

To do this we started creating a dedicated class to contain our return value, since the original GetNewsResponse contained directly the html that would be rendered in the website. In our class we needed instead the list of the news, that can be returned inside a JSON file in the HTTP response. So we defined our GetNewsForTeamsResponse class that contains a list of our NewsForTeams class. In this last data structure, we store all the information needed on our Teams application to show the news: the title, the text, an optional image, and other useful information such as the publication date, the number of likes and the link to the news, so that we can keep a sort of continuity with the original web portal. In this way, with just a click on the news rendered inside Teams we can redirect the user to the web browser, so that he can see the news directly on the intranet, if we want. In this way we keep the two as alternative, not substituting the old interface with the new one.

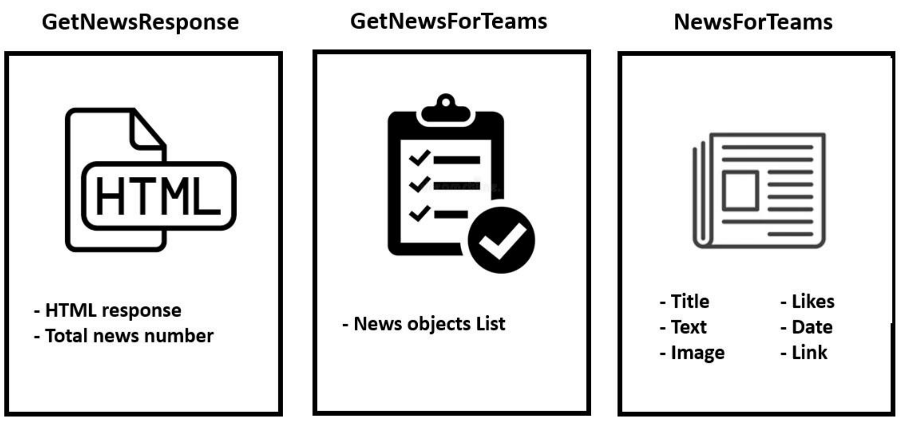


Figure 5.4 The different types of Objects used

After defining our classes, we needed to define our chain of calls, that can build the list of news and send them as a response to the calling Teams application. In the GetNewsForTeams method we elaborated the information to build this response as we needed it. To obtain the information about the news we used two methods from the Entity Manager service, but for both of them we needed a small redesign. So we built the GetNewsTeams, which will return the fundamental information, and the GetNewsInstanceWithUser, which will give us information on secondary aspects, such as the social information. After obtaining this information of the requested news, we can build our response taking what we need. From the "news" object we can obtain the id, the title and the publication date; from the "news instance" we can obtain the text, the image, the URL and the social information.

```
public GetNewsForTeamsResponse GetNewsForTeams(string Filter, SortDirection Sort, int Page, int MaxItem, string
User)
        {
            try
            {
                GetNewsForTeamsResponse resp = new GetNewsForTeamsResponse();

                lock (lockObj)
                {
                    resp = new GetNewsForTeamsResponse();
                    resp.NewsList = new List<NewsForTeams>();

                    User currUser = EntityManager.GetUserByEmail(User);

                    int NewsTotalNumber;
                    List<News> newsList = EntityManager.GetNewsTeams(Page, Sort, Filter, out NewsTotalNumber,
currUser);


                    foreach (News tempNews in newsList)
                    {
                        NewsForTeams tempNewsForTeams = new NewsForTeams();
                        tempNewsForTeams.Id = tempNews.IdNews;
                        tempNewsForTeams.Title = tempNews.Title == null ? "Titolo Nullo" : tempNews.Title;

                        tempNewsForTeams.PublishDate = tempNews.PublishDate;
                        try{
                            System.Text.StringBuilder sb = new System.Text.StringBuilder();
                            NewsInstance nIstance = new NewsInstance();

                            nIstance = EntityManager.GetNewsInstanceWithUser(tempNews, currUser);
                            if (nIstance != null)
                            {
                                tempNewsForTeams.Text = nIstance.ShortText == null ? "No text" :
nIstance.ShortText;
                                byte[] imageArray = nIstance.ImgUrl == null ? null :
System.IO.File.ReadAllBytes(nIstance.ImgUrl);
                                tempNewsForTeams.Image = imageArray == null ? "No image" :
Convert.ToBase64String(imageArray);
                                tempNewsForTeams.LinkUrl = nIstance.LinkUrl;
                            }

                            if (tempNews.FlagSocial)
                            {
                                GetLikeRequestWithUser newsLikeRequest = new GetLikeRequestWithUser();
                                newsLikeRequest.url = new List<string>();
                                newsLikeRequest.url.Add(nIstance.LinkUrl);
                                newsLikeRequest.user = User;
                                LikeService likeService = new LikeService();
                                GetLikeResponse newsLike = likeService.GetLikeWithUser(newsLikeRequest);
                                tempNewsForTeams.NLike = newsLike.NLike[0];
                            }
                        }
                        catch (Exception ex)
                        {
                            LogHelper.LogError(ex, "ERROR BUILDING NEWS", tempNews.NewsInstance.LinkUrl);
                        }

                        resp.NewsList.Add(tempNewsForTeams);
                    }
                }
                return resp;
            }
            catch (Exception ex)
            {
                LogHelper.Log(ex, LogCategory.B2E, LogSeverity.Error, ex.Message);
                throw;
            }
        }
```

For both the two methods for the news and the news instance, we operated a redesign from two original endpoints, and in both of them we inserted the current user passed as a parameter. This is due to the fact that we must retrieve the user with a dedicated method. This method is the GetUserByEmail, that will receive a string and retrieve the User object with all its information.

```
public static User GetUserByEmail(string userLogin)
        {
            //userlogin can be domain\\username or username
            User output = null;
            string uid = userLogin.Split(new char[] { '@' }, StringSplitOptions.RemoveEmptyEntries)[0];

            if (!String.IsNullOrEmpty(userLogin))
            {
                try
                {
                    if (output == null)
                    {
                        output = CacheManager.getObject(EntityType.User, userLogin) as User;
                        if (output == null)
                        {
                            // if not in cache load from DB.
                            using (DBHelper dbHelper = new DBHelper())
                            {
                                output = dbHelper.GetUser(uid);
                            }

                            // get field from sharepoint
                            if (output != null)
                            {
                                CacheManager.addObject(EntityType.User, output.UserLogin, output);
                                CacheManager.addObject(EntityType.User, output.Uid, output);
                            }
                            else
                            {
                                // assign a default profile
                                output = new User();
                                output.Name = userLogin;
                                output.Surname = String.Empty;
                                output.Uid = uid;
                                output.IsServiceUser = true;
                            }

                        }
                    }
                }
                catch (Exception ex)
                {
                    LogHelper.Log(ex.ToString());
                }
            }
            return output;
        }
```

Having this User object, we can use it in our personalized endpoints in the Entity Manage, GetNewsTeams and GetNewsInstanceWithUser, to reconnect our flow to the already existing chain of calls. In this way, just personalizing a three methods on the intranet portal, and creating a bunch of dedicated class, we obtained as a result that calling an endpoint from our Teams application we can retrieve a JSON file containing all the needed information about the requested news.

```
{
    "NewsList": [
        {
            "Id": 175,
            "Image": "No image",
            "LinkUrl": "/en/Pages/news/news2.aspx",
            "NLike": 2,
            "PublishDate": "/Date(1479976911507+0100)/",
            "Text": "test 12 short text homeless",
            "Title": "news2"
        },
       …
   ]
}
```

Figure 5.5 an Example of the JSON file containing the news

So, now we have an alternative flow of information that will take the user logged in Microsoft Teams and communicating with the intranet portal will obtain the needed data about the user and then retrieve to our external application the JSON containing the information on the news. With that, the application integrated in Teams will render the news in the dedicated tab.

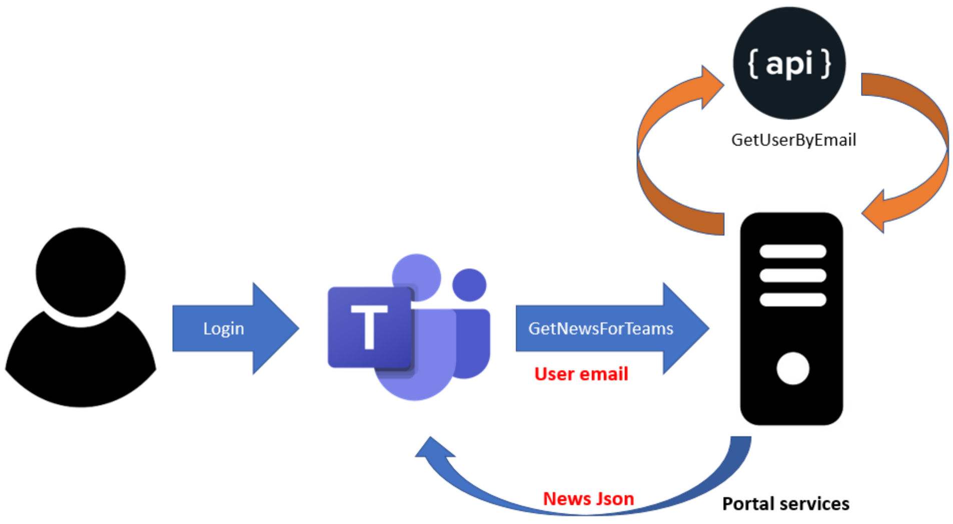


Figure 5.6 Retrieving the news in the Intranet Portal

5.3 “Like” as in modern social network

At this point, in our Teams application we have all the information needed to build a page that can replicate the homepage of the web portal inside Microsoft Teams. We closed the circle, for the moment: we realized a version of the homepage of the web portal that is accessible from withing Teams. At this point we can focus on some secondary functionalities that we wanted to integrate in the app, such as the “social-like” aspect.

In a similar way to what we have done with the news, we have to replicate the existing endpoints to avoid the problem of the user not existing in the context. So we defined also the AddLikeFromTeams method inside the LikeService of the intranet existing portal, that similarly to what we have for the news must contain also the information about the user. Also here we can call the GetUserByEmail method to retrieve the User object related to the calling user, but in this case we cannot reconnect to existing flow to obtain the information about the likes of the news.

In fact, to retrieve the information regarding the like of the news we needed also to build a fictitious token, that was needed at a deeper level in the chain of calls in the existing web portal. After the EntityManager, we needed to call the SocialHelper that would take the token regarding the logged user again from the context.

```
public static bool AddLike(Uri url)
        {
            try
            {
                Uri absUri = url;
                string connectionstring = string.Empty;

                // normalize url
                LogHelper.Log("url - " + absUri.ToString());
                if (!url.IsAbsoluteUri)
                {
                    string webAppUrl =
SPContext.Current.Site.WebApplication.GetResponseUri(SPUrlZone.Default).AbsoluteUri;
                    string absUriString = webAppUrl.TrimEnd('/') + url.ToString();
                    absUri = new Uri(absUriString);
                }

                connectionstring = ConfigurationManager.ConnectionStrings["SocialConn"].ToString();
                SPSecurity.RunWithElevatedPrivileges(delegate()
                {
                    Read.EnQueueLike el = new Read.EnQueueLike();

                    el.EnqueueOperation(DataContract.SocialAction.ADD,
                        absUri.ToString(),
                        EntityManager.UserTokenBinary,
                        SocialHelper.LikeGuid,
                        SocialMethods.AddTag,
                        new object[] { PartitionID, CorrelationID, connectionstring,
EntityManager.CurrentUser.UserProfile.RecordId, EntityManager.CurrentUser.UserProfile.ID});
                });

                return true;
            }
            catch (Exception ex)
            {
                LogHelper.Log(ex, LogCategory.B2E, LogSeverity.Error, "Unable to add like");
                return false;
            }
        }
```

As we can see from this block of code, that shows us the existing AddLike method inside the SocialHelper, inside the function that will enqueue the operation of adding the like is required a UserTokenBinary, that will be taken from the context, passing for the EntityManager.

```
public static byte[] UserTokenBinary
        {
            get
            {
                return SPContext.Current.Web.CurrentUser.UserToken.BinaryToken;
            }
        }
```

Again, we cannot take this token from the context, so we had to find another workaround, creating it on the fly for our needs. So, we defined also a custom AddLikeFromTeams inside the SocialHelper that generates inside it the needed token.

```
public static bool AddLikeFromTeams(Uri url, User CurrentUser)
        {
            try
            {
                Uri absUri = url;
                string connectionstring = string.Empty;
                LogHelper.Log("url - " + absUri.ToString());
                if (!url.IsAbsoluteUri)
                {
                    string webAppUrl =
SPContext.Current.Site.WebApplication.GetResponseUri(SPUrlZone.Default).AbsoluteUri;
                    string absUriString = webAppUrl.TrimEnd('/') + url.ToString();
                    absUri = new Uri(absUriString);
                }

                connectionstring =
ConfigurationManager.ConnectionStrings["SocialConn"].ToString();


                SPSecurity.RunWithElevatedPrivileges(delegate()
                {
                    using (SPSite site = new SPSite(siteUrl))
                    {
                        using (SPWeb web = site.OpenWeb())
                        {
                            var userName = "B2E\\" + CurrentUser.UserLogin;
                            var token = web.AllUsers[userName].UserToken.BinaryToken;
                            Read.EnQueueLike el = new Read.EnQueueLike();

                            el.EnqueueOperation(DataContract.SocialAction.ADD,
                                absUri.ToString(),
                                token,
                                SocialHelper.LikeGuid,
                                SocialMethods.AddTag,
                                new object[] { PartitionID, CorrelationID, connectionstring,
CurrentUser.UserLogin, CurrentUser.UserLogin });
                        }
                    }
                });
                return true;
            }
            catch (Exception ex)
            {
                LogHelper.Log(ex, LogCategory.B2E, LogSeverity.Error, "Unable to add like");
                return false;
            }
        }
```

Using siteUrl global variable we defined the SPsite manually and then using the username of our user we could obtain his token and go on with the enqueue operation that will add a like.

At this point, the external application is integrated in Teams and working, it is showing the news as in the intranet portal homepage and giving us the possibility to add and remove like from there.

5.4 The other solution: cloud workflow

With this solution we thought we could get the concept of integration even further. While in the first case we created a web application that is showed as a tab inside Teams, with our graphical interface, with the news rendered in our web page, as external object with respect to Teams, in this second case we want to use a different approach.

We thought about creating the news inside Teams as automatically generated posts on a dedicate channel. As we explained in previous chapters, Microsoft Azure offers a specific tool to create personalized workflow, using triggers and action: the Logic Apps. For our goal, another efficient and simple solution that fit our needs was to build a logic app. We should trigger it during publication of a news on the intranet portal so that it can send out that news in real time also to a selected channel.

In this second solution, we also thought to a way to render the news with Teams proper graphical interface. To realize it and show them as posts, we used the Adaptive Cards.

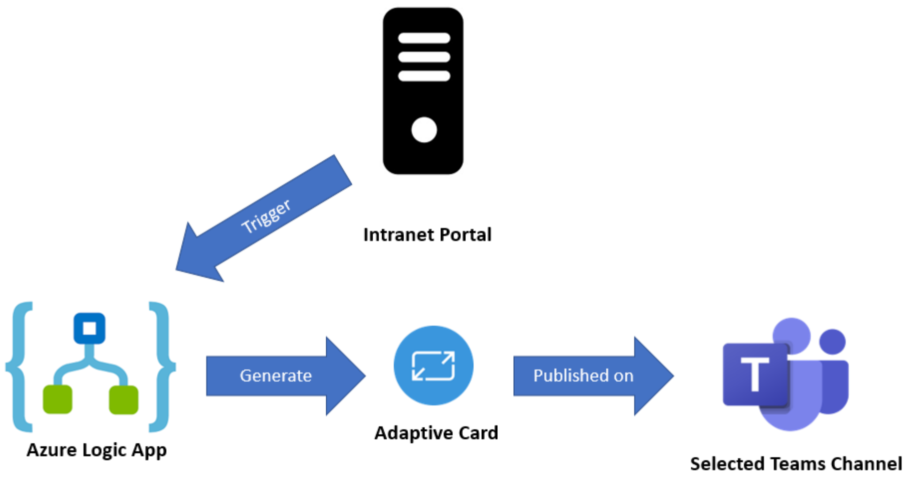
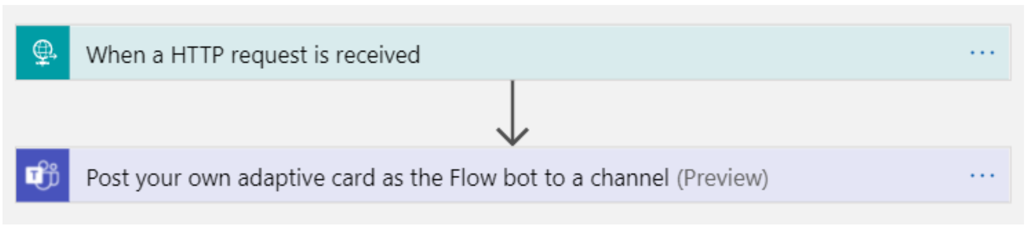


Figure 5.7 Publication to Teams with automatic workflow

To obtain this result we needed to create a Logic App in the Azure environment. Our workflow will be quite simple, actually: we just need to connect a trigger to an action. So, basically we have one trigger, that will activate after receiving an HTTP POST, and one action, that will send the Adaptive Card to Teams.

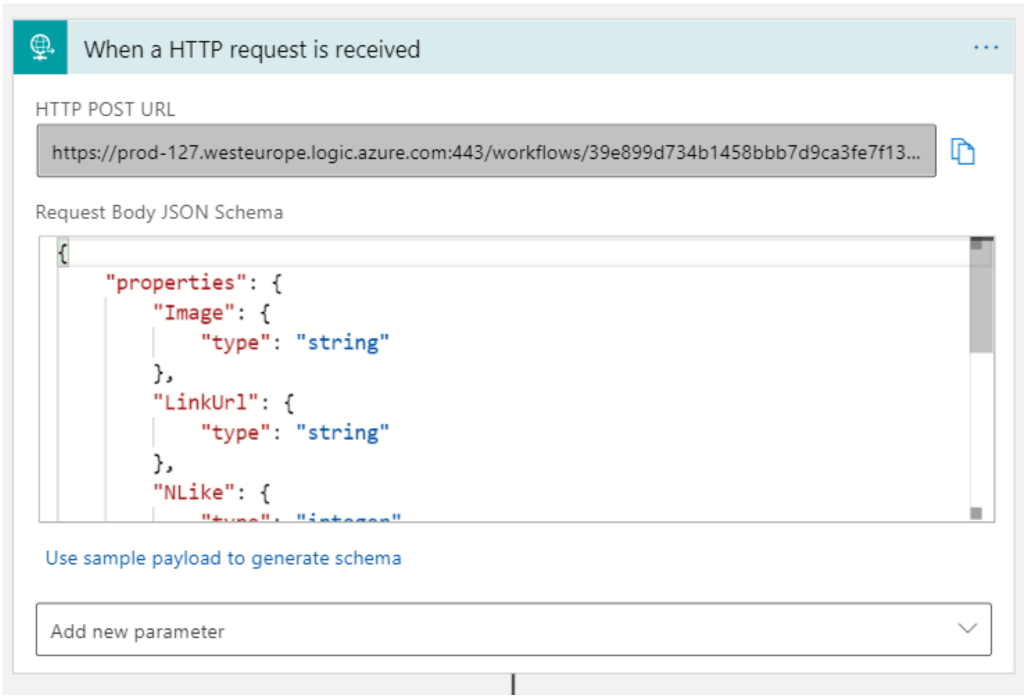


When a HTTP request is received ···

Post your own adaptive card as the Flow bot to a channel (Preview) ···

Figure 5.8 Schema of our Logic App

The simplest way to activate our Logic App from the intranet portal is with a HTTP call containing the information about the news to publish on Teams. So, we can define this trigger expecting a JSON containing the information about the news, such as the one we already were using to retrieve the data to our application.


When a HTTP request is received

HTTP POST URL

https://prod-127.westeurope.logic.azure.com:443/workflows/39e899d734b1458bbb7d9ca3fe7f13...

Request Body JSON Schema

```
{
    "properties": {
        "Image": {
            "type": "string"
        },
        "LinkUrl": {
            "type": "string"
        },
        "NLike": {
            "type": "integer"
```

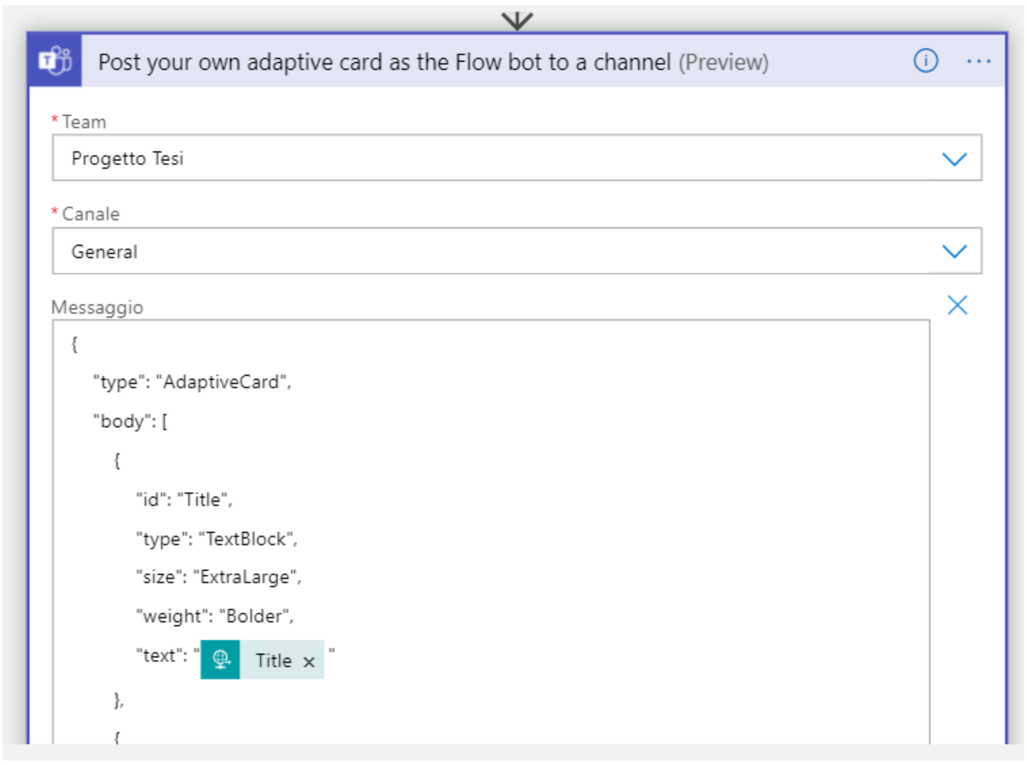
Figure 5.9 Definition of the trigger in Logic App Designer

As we can see from the figure, we can define an endpoint for the HTTP POST call and a schema for the expected JSON body, that as a class in a programming language state what properties will be expected. This schema in the end is basically another JSON file, that defines the parameters and their types. As we can see, we defined it very similar to the one sent back from the intranet portal to our Teams application, because it will activate this trigger exploiting the same data class used to return that news.

```
{
    "properties": {
        "Image": { "type": "string" },
        "LinkUrl": { "type": "string" },
        "NLike": { "type": "integer" },
        "PublishDate": { "type": "string" },
        "Text": { "type": "string" },
        "Title": { "type": "string" }
    },
    "type": "object"
}
```

In the end we can simply call that endpoint in a POST request including the news in the body, and like that we have triggered our logic app. At this point, we need to define an action that can send out the news to our selected channel in Microsoft Teams. The Logic App actions came in help in this case! In fact, Azure gives as the chance to define an action that will post an adaptive card to Teams as the flow bot. It gives us the possibility to choose the team and the channel where to publish the post, and then to define a message, as an Adaptive Card obviously.

What is interesting is that the Logic App designer provides to the developer an autocomplete functionality, that designing the adaptive card suggest to the user the dynamic content to use in it, based on the schema defined in the trigger. So, while building our adaptive card we will be suggested with the values of the title, image, link and others that we defined in the first block of the app. A powerful instrument that helped us filling our adaptive card with the data coming from the intranet portal.


Post your own adaptive card as the Flow bot to a channel (Preview)

* Team: Progetto Tesi

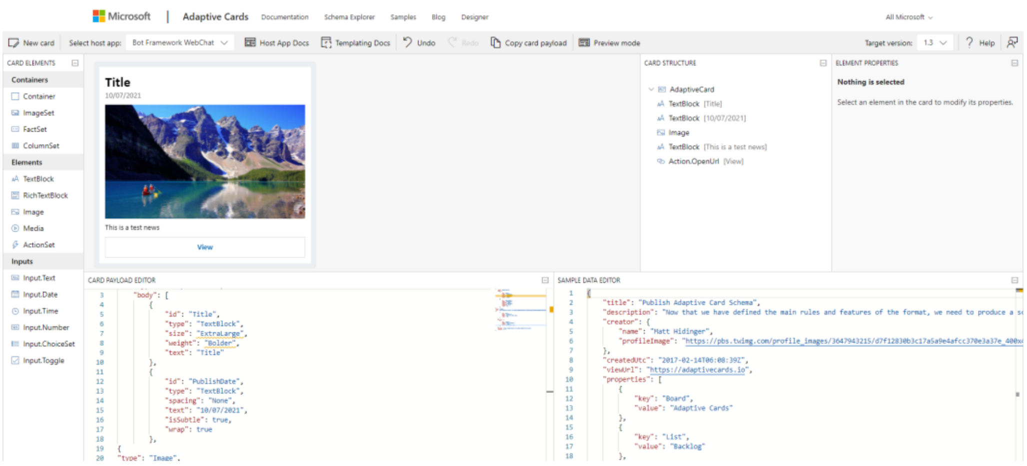
* Canale: General

Messaggio

```
{
    "type": "AdaptiveCard",
    "body": [
        {
            "id": "Title",
            "type": "TextBlock",
            "size": "ExtraLarge",
            "weight": "Bolder",
            "text": "Title"
        },
        {
```

Figure 5.10 Definition of the action in the Logic App designer

To define the adaptive card we used the dedicated designer, provided by Microsoft. With it is possible to test different version and see the different possible output that will result, also seeing how it will look in the possible host app, such as Microsoft Teams, both in light and dark theme. This designer allows to work with the code, writing down directly the JSON that will define the card, or exploiting its graphical interface, drag and dropping the component in the dedicated box, and then writing directly inside them.



```
    "body": [
        {
            "id": "Title",
            "type": "TextBlock",
            "size": "ExtraLarge",
            "weight": "Bolder",
            "text": "Title"
        },
        {
            "id": "PublishDate",
            "type": "TextBlock",
            "spacing": "None",
            "text": "10/07/2021",
            "isSubtle": true,
            "wrap": true
        },
    {
    "type": "Image",
```

```
{
    "title": "Publish Adaptive Card Schema",
    "description": "Now that we have defined the main rules and features of the format, we need to produce a s
    "creator": {
        "name": "Matt Hidinger",
        "profileImage": "https://pbs.twimg.com/profile_images/3647943215/d7f12830b3c17a5a9e4afcc370e3a37e_400x
    },
    "createdUtc": "2017-02-14T06:08:39Z",
    "viewUrl": "https://adaptivecards.io",
    "properties": [
        {
            "key": "Board",
            "value": "Adaptive Cards"
        },
        {
            "key": "List",
            "value": "Backlog"
        },
```

With this instrument we defined an example of the card how we wanted it to look, and then we transferred it inside the action in our Logic App designer. As we can see from the JSON file below, we can reference the properties of the trigger that preceded the action with the code triggerBody that makes simpler to build the response.

```json
{
    "type": "AdaptiveCard",
    "body": [
        {
            "id": "Title",
            "type": "TextBlock",
            "size": "ExtraLarge",
            "weight": "Bolder",
            "text": "@{triggerBody()?['Title']}"
        },
        {
            "id": "PublishDate",
            "type": "TextBlock",
            "spacing": "None",
            "text": "@{triggerBody()?['PublishDate']}",
            "isSubtle": true,
            "wrap": true
        },
{
"type": "Image",
"url": "data:image/gif;base64,@{triggerBody()?['Image']}"
},
        {
            "id": "Text",
            "type": "TextBlock",
            "text": "@{triggerBody()?['Text']}",
            "wrap": true
        }
    ],
    "actions": [
        {
            "id": "LinkUrl",
            "type": "Action.OpenUrl",
            "title": "View",
            "url": "/@{triggerBody()?['LinkUrl']}"
        }
    ],
    "$schema": "http://adaptivecards.io/schemas/adaptive-card.json",
    "version": "1.2"
}
```

At this point we have finished the development of our automatic workflow, and we can test it by triggering it, calling that endpoint with an external call.

Now, we must create a connection between the existing intranet portal and our Logic App. Clearly, we have to think something different, because in this case we do not have to send each time the block of news that must be seen in home page, since what will be published in the Teams channel will remain there. We need instead to trigger the app on publication of each news.

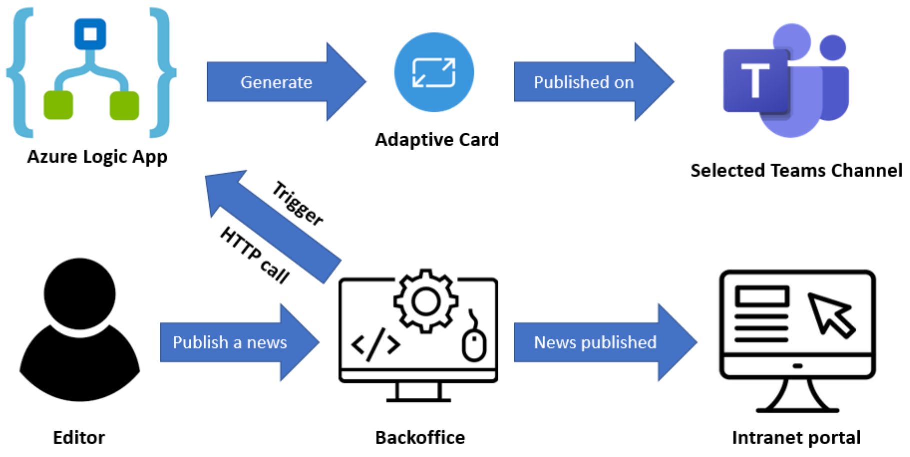


Figure 5.11 The publication of a news

So, we have to modify the existing endpoint that manages it in the existing portal, so that we can add a parallel mechanism that will create a JSON and call the logic app endpoint in POST with the news in the body. In this case we don't have any problem of authentication, we just need to add a piece of code to the existing method that adds this mechanism.

```
        static object publishNewsLock = new object();
        public PublishNewsResponse PublishNews(string Url)
        {
            PublishNewsResponse response = new PublishNewsResponse();
            try
            {
                lock (publishNewsLock)
                {
                    // Get News from page.
                    News news = EntityManager.GetNewsFromPage(Guid.Empty,
Guid.Empty, Guid.Empty, 0, Url);
                    // check if news exist in db
                    // if(exist)

                    if (EntityManager.ExistsNews(news))
                    {
                        news.FlagDeleted = false;
                        EntityManager.UpdateNews(news);
                    }
                    else
                        EntityManager.InsertNews(news);

                    /*** New mechanism ***/
                    string postUrl = "https://prod-
127.westeurope.logic.azure.com:443/workflows/39...";

                    news.PublishDate = news.PublishDate.Date;
                    string singleNewsJson = JsonConvert.SerializeObject(news);

                    var httpWebRequest =
(HttpWebRequest)WebRequest.Create(postUrl);
                    httpWebRequest.ContentType = "application/json";
                    httpWebRequest.Method = "POST";

                    using (var streamWriter = new
StreamWriter(httpWebRequest.GetRequestStream()))
                    {
                        streamWriter.Write(singleNewsJson);
                    }

                    var httpResponse =
(HttpWebResponse)httpWebRequest.GetResponse();
                    using (var streamReader = new
StreamReader(httpResponse.GetResponseStream()))
                    {
                        var result = streamReader.ReadToEnd();
                    }
                    /*** End of new mechanism ***/

                }
                response.Status = 0;
            }
            catch (Exception ex)
            {
                response.Status = -1;
                response.ErrMsg = ex.ToString();
                LogHelper.Log(ex, LogCategory.B2E, LogSeverity.Error, "PublishNews
failed");
            }
            return response;
        }
```

So, we basically just need to build a JSON from the single news that is being published and then call the endpoint of the Logic App with a HTTP POST that has it in the body. This will automatically trigger our workflow and make the news appear also as a post in our Teams channel. In this way we basically created an alternative to the integrated app.

Chapter 6

Result

6.1 Familiar for editors, familiar for users

The goal of this project was to emphasize the importance of tools that can ease the work life of employees in a context of always working from home, due to COVID or to its impact. As we said, working from home is something that will remain even after, and because of that we must think to solutions that can improve this reality. For this reason, we took an old web-based portal used to keep employees up to date on the company's news and decided to realize a quicker and simpler to use alternative.

Our goal was to make the change invisible for editors of the company, and we managed to do it. In fact, the interface to publish will not be changed; both local and global communication teams of the company will continue to take care of the news in the exact same way as they did in the past, nothing will change from that point of view.

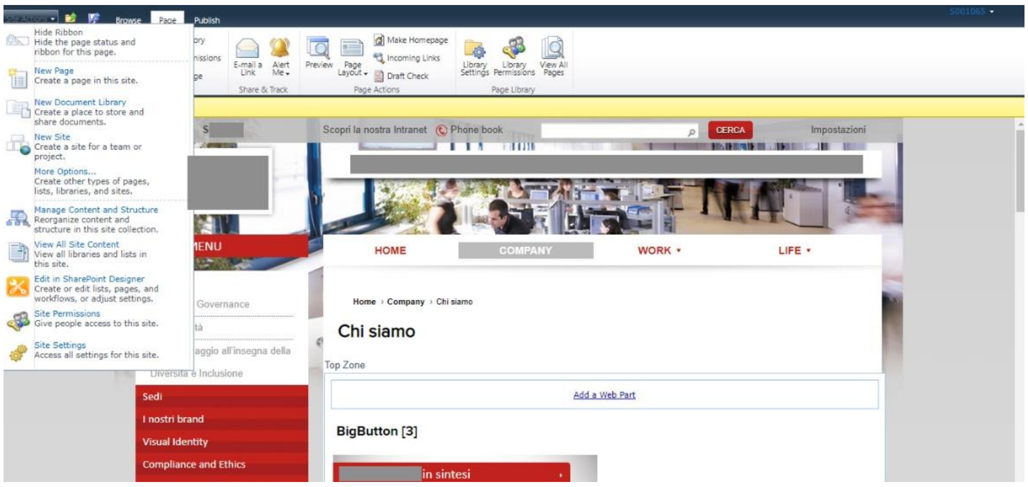
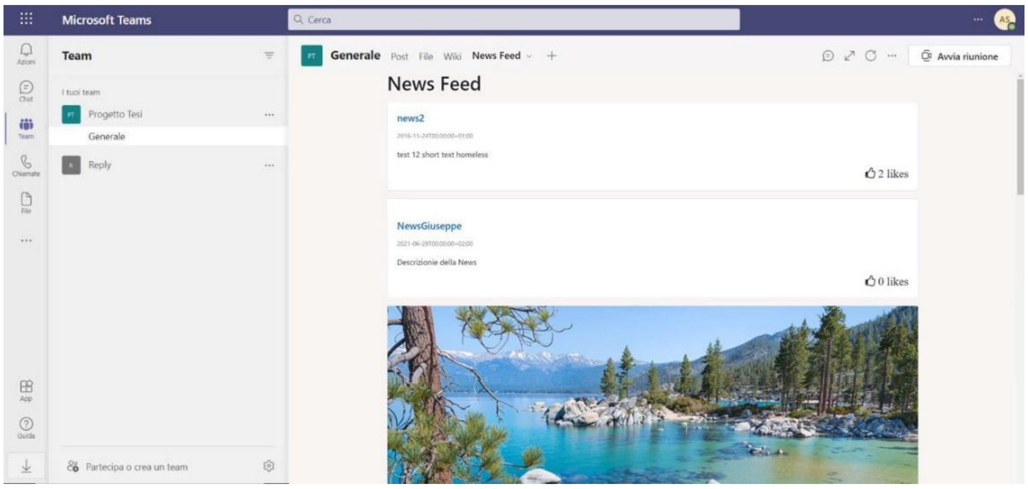

Browse
Page
Publish
Hide Ribbon
Hide the page status and ribbon for this page.
New Page
Create a page in this site.
New Document Library
Create a place to store and share documents.
New Site
Create a site for a team or project.
More Options...
Create other types of pages, lists, libraries, and sites.
Manage Content and Structure
Reorganize content and structure in this site collection.
View All Site Content
View all libraries and lists in this site.
Edit in SharePoint Designer
Create or edit lists, pages, and workflows, or adjust settings.
Site Permissions
Give people access to this site.
Site Settings
Access all settings for this site.
E-mail a Link
Alert Me
Share & Track
Preview
Page Layout
Make Homepage
Incoming Links
Draft Check
Page Actions
Library Settings
Library Permissions
View All Pages
Page Library
Scopri la nostra Intranet
Phone book
CERCA
Impostazioni
MENU
HOME
COMPANY
WORK
LIFE
Governance
aggio all'insegna della
Diversità e Inclusione
Sedi
I nostri brand
Visual Identity
Compliance and Ethics
Home › Company › Chi siamo
Chi siamo
Top Zone
Add a Web Part
BigButton [3]
in sintesi

Figure 6.1 Editors' interface of the portal

What instead will change is the final user point of view. While the user in the past used to access to a web portal from a browser, needing an extra login and some loading time, with the new configuration will not have to do anything, just log in the same platform used every day to communicate with its colleagues. There they will find also a dedicated tab or a dedicated channel, that shows all the same news of the web portal. The company managers will have the possibility to use the news interface in each team or channel where they want to show it.

6.2 Pros and cons of the two solutions

In the first solution, the users of the teams that access this tab will automatically see a personalized screen, as the old homepage of the intranet portal was. In fact, each user will be recognized by the account email, and the news showed will be based on the audience they belong to. We created an alternative to the old news system that basically works in the same way, but that is easier to access and consult.



News Feed

news2
2016-11-24T00:00:00+01:00
test 12 short text homeless
2 likes

NewsGiuseppe
2021-06-29T00:00:00+02:00
Descrizionie della News
0 likes

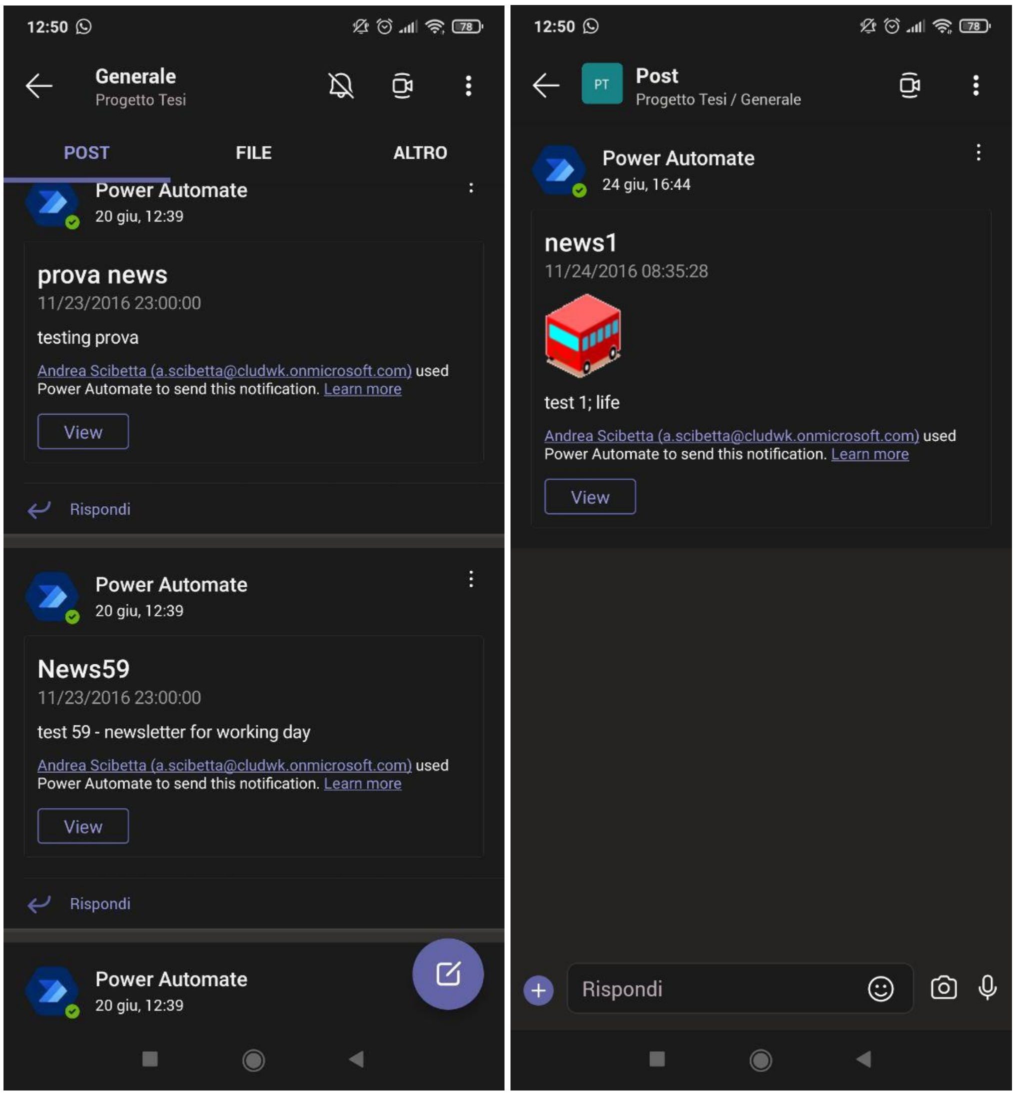
Figure 6.2 The News Feed Teams App

On one hand we have the first solution, that is clearly a transposition of the existing web portal homepage; on the other hand, we have the second alternative that instead work in a slightly different way. In that case we choose the channel where our logic app will post the news, and they will be published there in real time after the publication on the web site.

In this second possibility, the effect is of a perfectly integrated flow of Teams post, self-generated, with the possibility to add reaction and comment, and sharable on private chat. Clearly a solution that is distant from the original portal interface, but that can encourage a different way of fruition of the news system. With this implementation the notification system of Teams will be exploited to notify the user about new posts in the dedicated tab or channel.

Regarding this second solution, it would be more difficult to provide the audience system. It would require differentiating the channels where to post the news in form of Adaptive Cards from our automated workflow. We should create many channels related to the different users and publish each news on the proper ones, and in parallel allow the proper users to see the proper channels. Basically, it means to almost recreate the audience system from a different perspective and with different technologies.

Two different solutions that are both easier and quicker to use with respect to the original one, as we wanted, but also portable. The old intranet portal was only accessible by the private network of the company, and so not simple to reach from the external and in particular was very uncomfortable to use from mobile devices. With a solution based on Microsoft Teams instead, we can consult the news information from anywhere and from any device, such as a mobile phone or a tablet.


12:50

Generale
Progetto Tesi

POST FILE ALTRO

Power Automate
20 giu, 12:39

prova news
11/23/2016 23:00:00
testing prova
Andrea Scibetta (a.scibetta@cludwk.onmicrosoft.com) used Power Automate to send this notification. Learn more
View

Rispondi

Power Automate
20 giu, 12:39

News59
11/23/2016 23:00:00
test 59 - newsletter for working day
Andrea Scibetta (a.scibetta@cludwk.onmicrosoft.com) used Power Automate to send this notification. Learn more
View

Rispondi

Power Automate
20 giu, 12:39

12:50

Post
Progetto Tesi / Generale

Power Automate
24 giu, 16:44

news1
11/24/2016 08:35:28
test 1; life
Andrea Scibetta (a.scibetta@cludwk.onmicrosoft.com) used Power Automate to send this notification. Learn more
View

Rispondi

Figure 6.3 Two views from mobile

Chapter 7

Conclusion

These two solutions follow two very different approaches, one more conservative while the other more progressive. Both changes the perspective of the employee that consult the news, while maintaining unchanged the point of view of editors that publish and manages them. Although in both cases the desired result is obtained, there are important differences between them.

The main advantage of the first solution is the repurposing of the same identical interface in an external environment. It can fulfil the desire of maintaining a coherence with the existing portal and it will automatically show the news based on the audience of the user. Its disadvantages are related to the extended modifications to the existing infrastructure, providing each functionality to an external application, and limited possibilities of interaction with other Teams services.

Regarding the second solution, with it is simpler to automate the publication of the news on Teams, and it provides a full integration of them with the communication platform functionalities, since they are proper posts. The main disadvantages are related to the profiling, since it requires the creation of many channels where to publish the news, and to management of the news after publication.

From this work, what is evident is that the fruition of the news from Teams is quicker and allow a different way of interacting with the news. This highlight that the integration of external services into communication platform could make the consultation more comfortable for final users. As we enriched our news infrastructure with the functionalities offered for Microsoft Teams for its propriate post, this kind of integration can actually provide interesting new features to web based system. Obviously, each case should be analyzed differently, to evaluate pros and cons of the two solutions presented here, if they are feasible and which of them can bring more advantages to the system we need to integrate.

Future research on this topic should focus on implementing this solution in real scenarios. During this work we developed two demonstrative proof of concept to demonstrate the feasibility and the implementation of the two designed projects. A real use of these two solutions, realize ad hoc on a vast scale to evaluate the improvements, would give more definitive and precise result on which one is better and how both affect everyday use of the news interface by users.

A real implementation of these solutions with a real test on companies' employees will give the possibility to realize some statistics and to analyze empirically the obtained results.

Bibliography

[1] https://www.sylamtechgroup.com/wp-content/uploads/2019/04/intro-video-conferencing-wp-engb.pdf

[2] R. Singh, S. Awasthi, August 16, 2020, *Updated Comparative Analysis on Video Conferencing Platforms- Zoom, Google Meet, Microsoft Teams, WebEx Teams and GoToMeetings*, CS Department, MSI, New Delhi, India

[3] L. Yang, S. Jaffe, D. Holtz, S. Suri, S. Sinha, J. Weston, C. Joyce, N. Shah, K. Sherman, CJ Lee, B. Hecht, J. Teevan, July 30, 2020, *How Work From Home Affects Collaboration: A Large-Scale Study of Information Workers in a Natural Experiment During COVID-19*, Microsoft Corporation, from https://www.microsoft.com/en-us/research/publication/how-work-from-home-affects-collaboration-a-large-scale-study-of-information-workers-in-a-natural-experiment-during-covid-19/

[4] N. Bloom, J. Liang, J. Roberts, Z. J. Ying, February 1, 2015, *Does Working from Home Work? Evidence from a Chinese Experiment*, The Quarterly Journal of Economics 130, 165-218, https://doi.org/10.1093/qje/qju032

[5] T. D. Allen, T. D. Golden, K. M. Shockley, October 2, 2015, *How Effective Is Telecommuting? Assessing the Status of Our Scientific Findings*, Psychol Sci Public Interest 16, 40–68, https://doi.org/10.1177/1529100615593273

[6] P. Choudhury, C. Foroughi, B. Larson, 2019, *Work-from-anywhere: The Productivity Effects of Geographic Flexibility*, SSRN Journal, https://doi.org/10.2139/ssrn.3494473

[7] https://siepr.stanford.edu/research/publications/how-working-home-works-out

[8] E. Stoeckli, F. Uebernickel, W. Brenner, 2018, *Exploring Affordances of Slack Integrations and Their Actualization Within Enterprises –Towards an Understanding of How Chatbots Create Value*, Proceedings of the 51st Hawaii International Conference on System Sciences, http://128.171.57.22/handle/10125/50142

[9] S. Limkar, S. Baser, Y. Jhamnani, P. Shinde, R. Jithu, P. Chinchmalatpure, 18 February 2020, *Chatbot: Integration of Applications Within Slack Channel*, International Conference on Intelligent Computing and Communication, Intelligent Computing and Communication, 551-558, https://link.springer.com/chapter/10.1007/978-981-15-1084-7_53

[10] P. Lowenthal , J. Borup, R. West, L. Archambault, February 18, 2021, *Thinking Beyond Zoom: Using Asynchronous Video to Maintain Connection and Engagement During the COVID-19 Pandemic*, Journal of Technology and

Teacher Education, 28(2), 383-391, https://www.learntechlib.org/primary/p/216192/

[11] https://www.channele2e.com/wp-content/uploads/2020/11/canalys-cloud-research.jpg

[12] https://www.bitmat.it/blog/internet/connettivita/marzo-2020-il-traffico-internet-globale-e-aumentato-del-30/

[13] https://www.npmjs.com/package/adaptivecards/v/2.7.0

[14] https://www.adenin.com/blog/adaptive-cards/

[15] https://docs.microsoft.com/en-us/microsoftteams/platform/get-started/get-started-dotnet-app-studio?tabs=AS

[16] https://docs.microsoft.com/en-us/microsoftteams/platform/concepts/build-and-test/app-studio-overview

[17] https://tomtalks.blog/2020/12/microsoft-teams-statistics/