PWAs as Means of Communication Between Municipalities and Visitors

Petar Zhivkov Petrov

Abstract – Mass tourism has impacted many communities around the world, bringing along both positives and negatives. While it often boosts the economy and creates job opportunities, it can also lower the local quality of life and strain the administrations, several of which struggle to balance between their resources and the large influx of visitors. The following document explores the implementation of a PWA (Progressive Web App), that has been designed to aid the tourism branch of a local government. While applied in the context of a specific municipality and purpose, the resulting app can be viewed as a template that leads itself to being extrapolated and adapted to other areas.

Keywords – PWA, City Management, Tourism, Adaptability, Scalability

Resum – El turisme de masses ha impactat a moltes comunitats arreu del món, tenint-ne efectes tant positius, com negatius. Tot i que sovint impulsa l'economia i crea oportunitats laborals, també perjudica la qualitat de vida local i presenta una càrrega sobre les administracions públiques, moltes de les quals troben difícil equilibrar entre els recursos disponibles i el gran nombre de visitants. El següent document examina la implementació d'una PWA (Progressive Web App) dissenyada per ajudar el departament de turisme d'un govern local. Tot i que fou aplicada en el context d'un municipi i propòsit específics, l'aplicació resultant es pot percebre com a una plantilla, la qual facilita la seva extrapolació i adaptació a altres àmbits.

Paraules clau – PWA, Gestió de ciutats, Turisme, Adaptabilitat, Escalabilitat

1 Introduction

The tourism industry has become vital for many service economies. Be it seasonal or year-round, not only cities, but entire countries depend on this activity to generate profits and remain financially stable. While tourism can stimulate economic growth, break cultural boundaries and help people connect, when done in excessive numbers, it usually results in increasingly diminishing (and detrimental) returns.

Mass tourism and overtourism [1] are key examples of the negative phenomena associated with an unregulated tourism sector. These practices can be summarized as the constant and large flow of visitors to specific destinations, enticed by the affordable prices catered to them, and unfortunately, often by the opportunity to act without restraint. Due to the dependence on tourism, it is not unusual for authorities to deliberately ignore the consequences of overtourism, such as the rising prices for the local population and the growing trend of incivility.

• *Contact E-mail:* 1530304@uab.cat
• *Project Advisor:* Remo Suppi Boldrito (Department of Computer Architecture and Operating Systems)
• *Academic Year: 2024-25*

While there are some administrations that indeed do resort to passiveness, it should be noted that a sizeable portion of these destinations consists of smaller towns, where the available resources are a limiting factor. These municipalities usually lack the budget, materials, and personnel to effectively tackle the problems associated with their overwhelming visitor-to-resident ratios.

A solution to this issue could lie within the adoption of technology, allowing administrations to increase the effectiveness of their actions. Even though many potential lines of action exist, this document will explore a more effective way to interact with visitors through the implementation of a PWA.

PWAs (**P**rogressive **W**eb **A**pps) are, in general terms, "browser-based web applications" [2]. They are hybridized software, where the programming languages involved are web-development-heavy (HTML, CSS, and JavaScript), while their final look and feel can also correspond to modern mobile apps.

The mixed nature of PWAs gives them many benefits, such as substantial compatibility with different systems/devices, achieved through a single development cycle (when compared to native apps), as well as the possibility of offering cached/offline usability (in contrast to web pages).

While valid, the aforementioned benefits can be challenged by the ability of native apps to offer deeper integration with the device's OS, and the ubiquity of web pages. Albeit not really a technical aspect, they can also experience hardships through the pressure exercised by large multinationals against the growth of PWAs [3].

The latter trait has a direct correlation with the universality of PWAs and their independent distribution, allowing them to bypass app stores completely and avoiding the need to pay fees to these services. These qualities make them especially attractive for small teams and experimental projects, who need to gauge the reaction to their products through a low-risk deployment with a sizeable reach.

1.1 Motivations

As explained in the previous section, and owing to the knowledge imparted throughout this degree, it has become apparent that many of the difficulties faced by public administrations can be mitigated via the mindful and tactical employment of technology.

Although there are numerous areas to be improved, the main focus of this document will fall on tourism, an economic sector that generated approximately 12% of Spain's GDP in 2023 [4]. The geographical scope of the study will be centred around the town of Lloret de Mar, due to it being both the author's home and a municipality affected by mass tourism (sporting around 30 tourists per local resident) [5].

Lloret de Mar appears to be the perfect candidate for a PWA implementation: a relatively small administration (with limited resources) which needs to reach a large amount of diverse individuals quickly (an easy and cost-effective distribution). To keep up with the volatile nature of tourism, the app should lend itself to being readily accessible and streamlined towards the visitors' needs. Currently, there are no available options that fill the proposed niche, with the only existing smart app being native and focused on locals [6].

1.2 Objectives

The main goal of this project is the creation of a software solution specialized around the touristic aspect of Lloret de Mar and consequently heavily aimed at visitors.

When finished, the final app should be easy to use with a clear and simple design. It should feature content of interest to tourists (such as key locations or details about important contacts), as well as some customization options, like accessibility settings or a language menu (within the realm of possibility).

Albeit an academic project, the funding concern should not be ignored – when building the software, the employed plug-ins, libraries, and other tools should be open-source and/or low-cost. This point reflects the actual considerations made by small administrations in real-life scenarios.

2 METHODOLOGY

As a result of the end users being the main focus, the work on this project is based on the Agile approach, in combination with user-centred design elements [7]. This design philosophy is supplemented by the core values of accessibility and sustainability, and easily integrates within the existing urban infrastructure. It includes four main lines of work:

- **Research**: Study the currently available products and document their strengths and weaknesses. Analyse the needs of the consumers and contrast them with the previous items.
- **Co-design**: Co-operate with both locals and travellers, involving them in the creation process and ensuring that their practical needs are satisfied. Understand and build upon their feedback.
- **Iterative Prototyping**: Set achievable goals in short cycles or sprints. Periodically test the functionality, and integrate test user opinion before moving to a newer cycle.
- **Scalability**: Design the software to be easily expandable, as well as adaptable to different use cases. The app should effortlessly conform to diverse scenarios, lending itself to be used as a template for a plethora of other tasks, ensuring a similar performance without an excessive cost increase.

3 DESIGN AND IMPLEMENTATION

This section will serve as a walk-through guide showing how the app is built. Ionic (Angular) will be the base framework used to create the proposed PWA. The development will happen within a virtualized Ubuntu environment (through Oracle's VirtualBox) and code editing will be done in Sublime Text.

To ensure that the workload is managed efficiently, the process will be broken down into smaller, well-defined tasks. The plan is to move linearly through the tasks, starting with the more complex (or essential) ones and moving on to the simpler (or optional) ones. Here is the main to-do list in that order:

- Create the base template and its corresponding items, such as tabs or sidebar menus.
- Populate these items with the app's main content, including elements like maps and recommendations.
- Translate (when possible) the content that the users will see and use a global language selector to give them a choice.
- Include miscellaneous options, like a light/dark mode switch and accessibility features.
- Customize the look of the program and improve its presentation, through the addition of a custom splash-screen, icons and other graphics.

The first two items in the list can be considered core functions, being the backbone and main selling points of the app. In the case of unforeseen events or unplanned time constraints, the software should include at least these two components. This also means that the latter items are more prone to radical or conceptual change, addition, and removal.

The development will be aided by tools for conceptual visualization, like diagrams or wireframes (particularly in Figma). On the other hand, the changes made to the code will be automatically versioned and organized through a GitHub repository [9], where these modifications will be easy to track and see.

It is important to note that the main resource employed to create the app is Ionic's official documentation [8]. A large swath of the program's features have been sourced directly through the aforementioned source, customized to the app's needs as seen fit. Reference-wise, the software's main inspiration comes from Ionic's Conference App, which was made in order to demonstrate many of Ionic's features [10].

3.1 App Structure

Before going into details about the application's format, it should be mentioned that when creating a new project, the Ionic framework gives the user three main templates: blank, tabs and side menu. While the blank template is obviously an empty slate for more experienced developers, the latter two, while differing visually, serve the same purpose – to create a compact and clean routing solution (seamless and easy transition between the app's functions).

As mentioned beforehand, the app's structure is based on the Ionic Team's Conference App. The Conference App features both tabs and a side menu, having them set in a way where they share the same actions, i.e. pressing the "X" tab will have the same result as pressing the "X" item in the menu. The shared outcome is redundant (due to it being an illustrative app), which is why this project's PWA implements them in a differentiated manner.

The tabs section features the main content of the app, incorporating the elements intended for the user to interact with the most: categorized maps, weather information or an event feed are some functionalities present in the tabs. The latter ones are split into four categories: 'Home' (default page, with sundry content), 'Explore' (map-related page), 'Enjoy' (event-related) and 'Get Help' (emergency directory). On the other hand, the side menu contains miscellaneous settings, such as a language selector or a dark mode toggle, which the user might only use occasionally. Before starting to develop the PWA, a mock-up was created (*Figure 1*) through Figma [11], to help visualize what the app could possibly look like. All final features are explored in more detail throughout the following subsections, in order of implementation.

3.2 Home Page

Following the creation of the app's basic structure, the 'Home' page is the next created feature. The project's home section is designed with simplicity in mind, offering the users a few quick functions, rather than mainstay content. It serves as the default tab, being the first one seen on launch.

As stated above, the 'Home' page does not accommodate too many elements, containing a small welcome message, weather information, and embedded elements: a social media frame and a live webcam webpage. While the welcome message is a simple stylized plain text solution, the weather feature combines Ionic's floating action button (fab) with iframes [12] sourced through the widgets provided by the

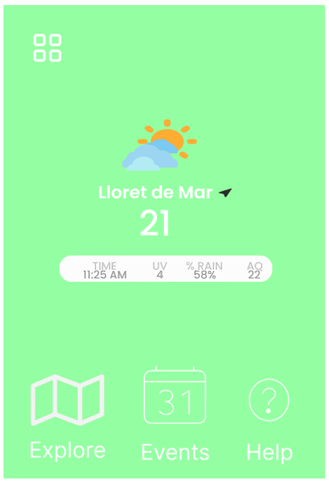


Fig. 1: A simple Figma draft, to help visualize the hypothetical look of the app, mirroring some of the Conference App's design choices.

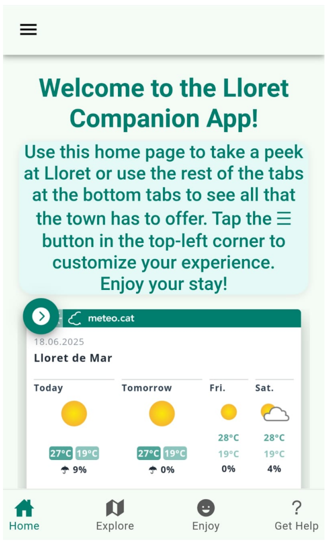
"Meteorological Service of Catalonia" [13]. When clicked, the fab button expands and shows four actions that the user can select from: a basic local forecast (shown by default), a detailed one, a rain radar of Catalonia and a map featuring weather alerts for a 48-hour period. On each press of one of the options, a single widget is displayed according to the current selection, to avoid the cluttering that would come with a simultaneous deployment of all the widgets. The fab buttons also store the user's last selected option, implemented through Ionic's storage module, applied again in some of the other features.

The next feature is a snippet of a social media feed. Originally, this embed was planned to contain either one or two "Twitter" (nowadays called "X") accounts that would be relevant to a visitor, such as those of the town hall or the local police department. This idea fell through due to changes to the company's API and embedding policies [14]. An alternative was found by embedding another social media – "Instagram". While restrictive (only displaying a few posts), it can serve as a subtle promotion device offering a glimpse of what the town has to offer, i.e. Lloret's tourism section's publications.

A live video feed was chosen as the final feature of the home section. The "Twitter" integration was tried again in the form of simple hyperlinks leading to an external browser. But the limitations mentioned before also apply to browsing without being a signed-in and verified, with quick rate limits and unsorted posts hindering the user experience. During this time, a feature of one of Lloret's digital newspapers, "Lloret Gaceta" [15], served as an inspiration for the video feed idea. Their web page features a section with an embedded video player, with live footage of the town's main beach. Upon further inspection through the browser's developer tools and the source code of the page, it was made clear that the video's source came from a company called What's Up Cams [16]. The latter houses several live streams of cities throughout the world.

Without finding a more attractive alternative, it was decided that the same source would be implemented in the app's 'Home' page. At first, the implementation of this element was tackled through the use of "Video.js" [17] and "hls.js" [18]. The attempted integration of these plug-ins was unsuccessful, resulting in a video player that never loads, owing to a failed manipulation of the video's .m3u8 file. To ensure that the idea would be realized, it was decided, begrudgingly, to add the whole web page as an embedded element. While rather simplistic and not as sleek as a lone video player, this compromise kept the desired core functionality.

The final look of the 'Home' page (*Figure 2*) takes inspiration from the initial Figma design. While the visual elements differ, and a button has been added for the home section, one can still see similarities between the layout of the two.


Welcome to the Lloret Companion App!

Use this home page to take a peek at Lloret or use the rest of the tabs at the bottom tabs to see all that the town has to offer. Tap the ☰ button in the top-left corner to customize your experience. Enjoy your stay!

Fig. 2: The app's final design, based on the initial Figma mock-up

3.3 Map Section

With the 'Home' page done, the next set of functions to be created were the 'Explore' tab's map-related ones, built through the Leaflet JS library [19]. Albeit unplanned, this part of the PWA ended being the largest by a noticeable margin, regarding both the amount of code written and the final number of features. Perhaps it is the similarity between the nature of the components and the knowledge and experience gained from this degree's curriculum, that led to its expansiveness (for the programming standards at hand).

The base layout consists of a title, followed by four buttons, with each one of those corresponding to a different map type. Once again taking inspiration from the Conference App, interacting with the map buttons opens a window with the chosen content, without routing the user to another page. This has been accomplished through one of Ionic's components – 'Modals'.

3.3.1 Modal

Per the Ionic documentation, "A Modal is a dialog that appears on top of the app's content, and must be dismissed by the app before interaction can resume…". When one tinkers with the provided examples, it becomes clear that this component is really versatile, and for a comparison of sorts, behaves like a floating independent Ionic tab. This idea was cemented through the conference app's usage of modals, where once open, these filled up their space akin to tabs, with the added benefit of avoiding visual clutter in the tab bar tabs.

Due to the envisioned extent of the modal, this component was generated, in contrast with the rest of the features present in the app. The purpose of such action being to avoid any clashes with the base tab, as well as to isolate any potential errors. Regarding the code, a reference to the modal was integrated inside the 'Explore' tab's files, so that the different buttons correspond and load different maps. This was accomplished through the use of events, which link the user's choice to its respective modal map code.

3.3.2 Map Types

When the user chooses a map, the modal displays a base map (*Figure 3*) and populates it accordingly. This map includes some basic leaflet elements, such as zoom control, layer control and making use of the bounding parameters of leaflet, limiting the area that one can navigate through. It has to be clarified that this is not a bug, but in fact intentional. Since the app's main focus is the town of Lloret de Mar, having the ability to view locations too far removed would render it pointless. The effect of this combination translates to returning the user to Lloret's general area, which also includes some of its surroundings.

The basic leaflet features have been expanded further with the inclusion of a search function [20], represented by a magnifying glass icon. This plug-in comes with a list of suggestions for searches sharing address keywords, as well as automatic zoom to the user's query. The search function was attempted through various other plug-ins, but the referenced one was the easiest to integrate and work with.

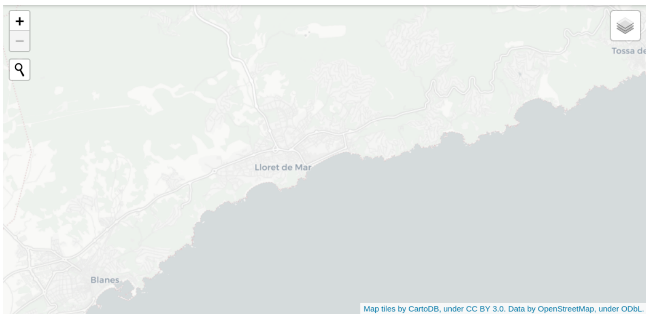

Tossa de
Lloret de Mar
Blanes
Map tiles by CartoDB, under CC BY 3.0. Data by OpenStreetMap, under ODbL.

Fig. 3: The base map shared by the different map types

With the base map ready, the logic behind the information that loads on it is done through a simple if-else-if loop, where the parsed choice of the user is the condition to activate one block instead of another. The first of these conditions is the 'Geolocate Me' one. This map type is fairly simple; its function being to determine the user's location and either pointing to it on the map with a marker, or displaying an error message if the geolocation fails. At first, the geolocation method employed was Leaflet's native one, but due to frequent errors it was swapped with code from a map type further down the development time – the Geolocation plug-in from the Awesome Cordova Plugins repository [21].

The next map type is called 'Points of Interest'. This map is populated by various leaflet objects, such as markers, circles and polygons, each one representing a place that is considered relevant to a potential visitor. It is important to mention that the way these data points are stored is inside a JSON data file [22]. JSON files are used to store JavaScript objects, which can then be easily read and interacted with. Their structure also allows for easy modifications, or even automated writing/erasure.

In the case of this POIs map, the objects within the JSON file were grouped by categories (Culture, Nature, Sports, . . .). Each individual object had properties defining its map object type, its geographic coordinates, the pop-up message associated with it and a value indicating its colour (more than one when it came to circles and polygons). Also, for the markers the icon used was sourced through the repository of the user "pointhi", allowing for differently coloured finishes of the default leaflet marker [23]. Inside the main script file, the JSON data was used in its respective function, added to the map and linked to a category selector which enables the user to choose which categories to display on the map.

The third type is the 'Path Guide' map, featuring the Awesome Cordova Plugins' map direction/navigation function and leaflet's 'Ant-Path'. These plug-ins enable automatic navigation between locations, complete with a representation of the route to follow and a pane that guides the user from their starting point to their destination. The initial version established the starting point through the geolocated position of the user's device, but was modified to be manually selected. Cordova's geolocation function was inserted into the 'Geolocate Me' map type instead.

The fourth and final map type of the PWA is called 'Safe Points', borrowing heavily from the 'Points of Interest' one. The reason for this to not be included in the POI was to avoid saturating the latter. However, the final 'Safe Points' map became too lacklustre, and a prime candidate for a possible overhaul. On the technical side of things, it is a simplified POI map, featuring a single leaflet object type – markers. These are once again grouped by categories and called from a JSON file. The script adds them to the base map and creates a pane that controls the displayed categories.

3.4 Event Section

The third tab is titled 'Enjoy', being a page that contains events and returns to a simpler object implementation, where the Ionic components are not generated, but implemented directly into the tab's code. When it comes to this page, the main feature comes from the 'Card' component, which is intended to display complex structures neatly. In the case of the Events section, the cards were grouped by categories and presented as vertical scrollable lists.

The way that the categories are displayed is meant to be intuitive, allowing the users to quickly scroll to the items they're interested in. The individual cards are compactly styled, starting (from left to right) with an illustrative image, a name and a brief description and ending with optional dates, locations, and Internet links. When the body of the card is pressed, an 'Alert' is opened, featuring an expanded description of the selected element. This could have been done through the use of modals, but it was considered excessive given the brevity and simplicity that a few string messages entail.

The established structure of the cards allows for the inclusion of multiple types of items, ranging from food establishments, to musical festivals and even cultural celebrations, like holidays or traditions. In the same fashion as some of the map types, the data source for the cards consisted of a JSON file. This way the data entries can be generated effortlessly without overburdening the main files with lines of repetitive code. The events were grouped, as usual, into categories and containing the values mentioned before and shown in the final card elements. This time the code was simpler, mainly featuring event controllers and the JSON's parsing.

3.5 Help Section

The last of the main tabs is the 'Get help' one. This page is straightforward and minimalistic, purposely built this way to ensure that the visitors can easily obtain information when they need it the most, i.e. when they require help.

Code-wise, the page is structured as a list, with a sizeable banner at the top reminding the users of the emergency number used in Spain. Concerning the list proper, its items contain a name, a brief description, an address, and a clickable web link. They also contain an interactive phone button, which passes a customizable telephone number to the device's calling program. The same method is used for the emergency number in the banner.

As was the case with the last two tabs, this information is stored inside a JSON file, allowing clearer codes and quicker modifications. Inside this data file, one can find the values pertaining to the information present inside the list's items, interpreted and applied correctly through the main code.

3.6 Side Menu and Miscellaneous

At this point in time, the application's main features have been created, tested, and are ready to launch. This implies that this project's primary goal has been achieved, meaning that the PWA is complete enough to enter the building and distribution stages. The remaining elements to be explored are secondary to the core components, and in a real-world scenario, their integration would not be critical to the application's release.

Following the order they were developed in, the first of these to be described will be the side menu. Although its basic appearance and behaviour were done quite early, programming each individual functionality came after the main tabs were done. The final set of optional features is composed of four buttons, one of each for: an accessibility menu, a language selector, a toggle between colour modes and a supplementary alert. The next subsections will be dedicated to the first three options. On the other hand, the alert is easily explainable: it functions as a brief information box, akin to the 'About' sections of many of today's apps and sites. It is presented as an alert dialog that features the PWA's name and a simple description about its nature.

3.6.1 Accessibility

Accessibility, in software, refers to the set of features that make programs easier to use for people with impairments.

Some of the more well-known accessibility tools are alternative palettes for colour-blindness, or text-to-speech synthesizers that can "read" strings and auxiliary information out loud.

During development, many choices were considered, and a few were even implemented, but the final result only includes a single option – a font size increase. Once the accessibility button has been pressed in the side menu, a dialog box shows up, presenting the user with a 'Toggle' component to change the font size. When pressed, most strings increase their size, and return to their initial shape when the toggle is disabled.

Code-wise, the box containing the accessibility options is actually a modal component. As opposed to the one used for the creation of the maps in the 'Explore' tab, this one is built inline; meaning that no component generation took place, integrating the whole element inside the pre-existing files of the side menu. The final shape was obtained through styling. Propagating the size change is done via a simple logical comparison, wherein the actuation of the toggle enables or disables a variable. When this variable is active, the app's global style sheet overrides key elements containing strings, amplifying their size values.

As a side-note, the features that did not make the final cut were a screen reader, a high contrast mode and a reduced animation feature. The tentative screen reader option was deemed too difficult to implement and was discarded early to avoid wasting development time and resources. The other options actually made it into the code and were functional, their results however were not satisfactory. The high contrast mode behaved like a subpar version of the dark mode, whereas the animation reduction lacked any apparent effect.

3.6.2 Dark Mode

Owing to the modern innovations related to eye care, such as night light and dark mode options, and featured in many of today's devices, an option that enables this functionality was regarded as an attractive addition to the app. Thanks to the Ionic documentation's example tutorial and code snippets, this utility had a smooth implementation.

Using the previous font size methods as a blueprint, the dark mode is built up in the same way. Concerning its appearance inside the side menu, it is represented by a label of the same name and a toggle. This toggle's actuation is linked to an event, which shares the state it is in, and can be used to change the appearance of the app's elements. To accomplish this change, the app's global style sheet contains predefined values and labels for the visual style of the darkened elements, sometimes also including light theme counterparts.

While most elements, like backgrounds and (Ionic) icons, changed without a hitch, others required tinkering and fine-tuning. The text inside alerts or the event and date information in the catalogue cards needed custom colours, otherwise they clashed with the dark background and were hard to read. There were also some conflicts present with some of the pages, due to their code explicitly declaring Ionic's light colour palette as their theme. This oversight was corrected by simply removing that property, which led to automatic colour switching that corresponded to the currently toggled mode.

3.6.3 Language Selector

The last side menu option enables the user to change the app's language, using a button fashioned from a 'Select' component. Actuating the select opens up a list of values one can choose from, in this case – languages. In the same vein as the past two features, the language is also listened to through event controllers, subsequently being stored and changed with the use of functions. But while the other two were completed mainly through SCSS styling, how does the app get translated? The answer lies within an Angular library: "ngx-translate" [24].

"Ngx-translate"'s expansive documentation and tutorials made the final result possible in a notably hassle-free manner. After loading up the necessary modules and correctly setting up the library's variables, keywords and functions, the library runs in a very simple manner: it parses the strings from JSON files and displays the appropriate ones depending on the currently selected language. In layman's terms, the proper text chunk is obtained from a collection of data.

This app uses three languages, which means that three different JSON documents had to be created manually, and placed inside a folder that is predefined by the 'translate' library. While the languages themselves are different (even including one that is not written in Latin, but uses the Cyrillic script instead), the JSON structure had to be identical between all language files. This results in files that share the same base objects and names, varying only in the values assigned to the names. This is not a hindrance; on the contrary, it eases the workload of having to manually check and translate every single word, and conveniently lends itself to automatization. The files were completed in the same manner: once the first one was done, it served as a template and was fed to a modern translation service, saving time and only needing manual adjustments when it came to a few mistranslations.

Upon the completion of the JSON files, the final step includes replacing the strings to be translated with an "ngx-translate" method that calls the keyword from the language file, displaying it in the currently selected language. This last step included having to refactor the previous tabs' JSONs, as well as needing to tweak some lines, but was overall rather straightforward. Once the lines were rendered correctly, the remaining errors (mostly orthographic) were easily seen and corrected by simply editing the language files.

3.6.4 Code Clean-up and Fixes

With the PWA's features done, this short section only serves to document the final touches done to the app before deployment. Most of these were related to styling and code-polishing tasks, such as giving a more coherent look, removing obsolete or leftover code, and commenting some key lines for future reference. This is also where the identifiable errors were tackled and fixed.

In relation to the tools that were used, aside from the main ones like Ionic's serve command (for generating a web page) and the general queries to forums and guides, the developer tools included in most modern browsers were heavily employed. They were often used to complement Ionic's own error detection, since the two differed more than once. Furthermore, the real-time CSS modifier helped to adjust the style without having to constantly rebuild the page, on top of allowing one to experiment with the visual elements. Lastly, the "Device Mode" tool allowed a preview of the PWA's behaviour when ran in different devices. This was especially helpful when it came to the next and final step of the project's implementation – deployment.

3.7 PWA Deployment

The last step of the project's development, and perhaps the most important one, is the deployment of the software. This means that the program will be shared with the world, and users can interact with it as intended. There are many ways to launch an Ionic (Angular) app, including publishing it to a third-party hosting service, self-hosting on one's own server, wrapping it as an APK to be installed on iOS or Android devices, among others.

For the PWA at hand, the chosen method was through GitHub's Pages hosting service [25], which neatly integrates the live link with the project's repository. When the page is opened, it behaves exactly as if it was created through the serve command, allowing the users full access to its contents and features. The hybrid nature of the Progressive Web Applications and its potential is fully apparent when deployed through this method; the page will respond correctly on both computer and mobile devices.

In addition to a standard website's set of features, the PWA offers new options. When browsed on a computer, the users can be offered a QR code that opens the app in their mobile device's browser. Furthermore, when viewed through the latter, the page can be added to the home screen of the device via shortcut. Opening the latter enables the PWA to emulate the looks of a native app, while running on the browser that created it.

4 Results

After successfully developing and deploying the app, this section will serve as a short overview of the final outcome, summarising how the features flow and showing its PWA-specific functionalities. A more detailed showcase, featuring screenshots from the final build, can be found as an appendix at the end of this document (*Appendix A.1*).

4.1 App Navigation Logic

The definitive version of the program behaves in a straightforward manner; after launch, the user is taken to the 'Home' page. From the latter, one can choose to navigate to either the app's side menu or any of the other pages. This freedom of choice persists throughout the app, since the side menu can be accessed from any page, and the 'Home' page can be returned to at any moment. The only element that breaks this rule is the 'Map Modal', the component that loads the maps for the 'Explore' page. When this particular modal is opened, one can solely go back to that page by design. A diagram has been created to better visualize this logic (*Figure 4*).

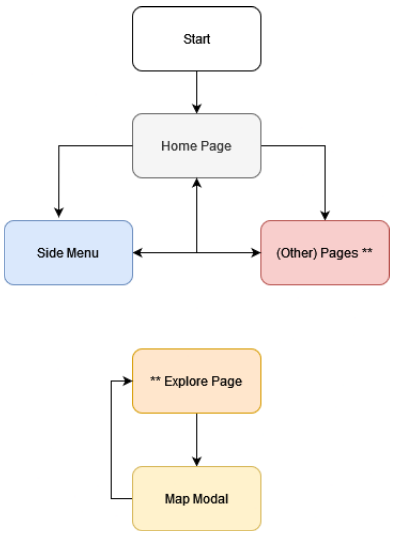
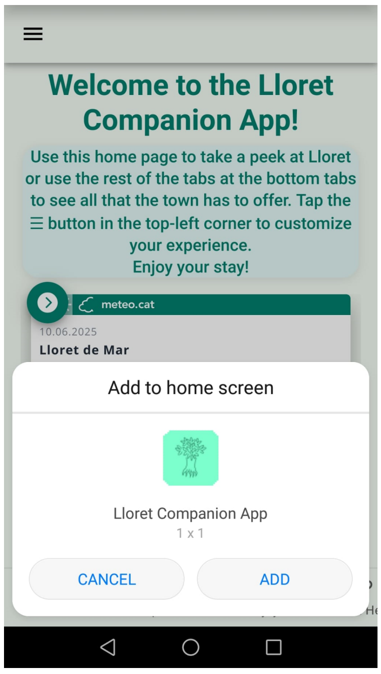


Fig. 4: Diagram representing the flow of the PWA

4.2 PWA-Exclusive Features

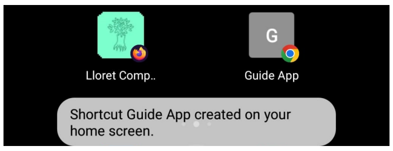
As has been established throughout this document, Progressive Web Apps combine elements from both websites and mobile applications. When launched, they behave as a regular web page, but thanks to being supported by most modern browsers, one can also make them behave like standalone programs without installing any software. In the case of mobile browsers, this function is often called "Add to Home Screen", or a minor variation of the latter. For desktop/laptop computers, some browsers offer the option to "install" the PWA, creating a direct shortcut that is opened through the same browser.

The following screenshots (*Figure 5* and *Figure 6*) show this process on a mobile device, highlighting the ability to set a custom name for the shortcut, which can help it stand out to the user. On the other hand, depending on the level of browser support, the final appearance of the icons might differ.


Welcome to the Lloret Companion App!

Use this home page to take a peek at Lloret or use the rest of the tabs at the bottom tabs to see all that the town has to offer. Tap the ☰ button in the top-left corner to customize your experience. Enjoy your stay!

Fig. 5: The PWA shortcut creation process



Shortcut Guide App created on your home screen.

Fig. 6: The resulting shortcuts, differing in their appearance. The one on the right features a customized label

5 Conclusion

Through its definitive version, the application has successfully achieved the objectives outlined at the start of this document. It is simple, with an easily understandable design, featuring content catered towards the town's visitors, filling a niche that is currently unexplored by the local administration. Furthermore, it accomplishes this by using completely open-source software and no funding.

This last point, along with its ease of development and distribution, make it a really attractive alternative to native apps, especially in the hands of a full-sized team with the backing of the local government. It has been intentionally designed with scalability in mind – being easy to modify according to the ever-changing needs of the market: swappable content, updatable events or even the addition of new languages; none of these upgrades require too many resources, while the returns can be as significant as attracting entirely new tourist demographics.

Finally, one should prioritise the lodging establishments, such as hotels, when reaching new tourists. As these businesses are often the first point of contact between the visitors and the town, providing them with the app from the outset could be beneficial to both their stay and the local economy. Since the app can be accessed as easily as scanning a QR code, strategically placing these could introduce the PWA to the tourists right at the beginning of their stay. There is also the possibility to offer the base app as a template, with each establishment modifying it as they see fit according to their particular needs.

6 Future Work

Although the final result is fully functional, there are still improvements to be made and additional features to be implemented. These prospective enhancements have been mentioned throughout the document, and can be classified into two categories:

- **User Involvement**: As described in the project's methodology, the development of the app should be done in tandem with user feedback. Albeit the final app is working correctly, it should be seen as a proof of concept, rather than a market-ready product. While it illustrates what a tourism-centered PWA could be like, it lacks the design stages involving the users.

  The first of these should consist of the participation of the interested parties before the public release of the app. This phase should include quick cycles of meetings, brainstorming and demonstrations, to ensure that the application is developing according to the real needs of the involved groups. Their experience and requirements can also serve to contrast the theoretical research, guaranteeing the launch of a software appropriate for the reality of the situation at hand.

  The other phase that should be addressed is the release to the consumers in a real-world scenario. Assuming that the local administration is the one backing up and deploying the PWA, the deployment phase could feature a plethora of betterments. These could be, amongst others, a better marketing campaign, self-hosting the web page, and monitoring the end-users' feedback, to ensure that any undetected errors during development are fixed quickly. Having access to a large volume of user reviews can also aid the development team in understanding the app's strengths and weaknesses, meaning that any potential changes can simultaneously refine the functionality of the software and meet the market's needs.

- **Software Additions**: Most of the technical improvements have already been mentioned in the Design and Implementation section. This fragment serves as a concise summary of these items.

  To begin with, even though the current application does not support offline functionality, that was not its intended purpose. It was designed to work while connected to the Internet, and it foregoes the benefits of offline capability for maximum loading speed. To achieve offline use, it could be wrapped as an APK. Another option could imply the usage of a caching solution, that contains preloaded data.

  Regarding the main features, the most lacklustre one is the Accessibility menu. As was mentioned before, a text-to-speech function would be the prime candidate to be added in the case of a revision. To a lesser extent, there could also be a chance to implement colourblindness and proper high-contrast modes. The rest of the sections are generally complete, lacking some small refinements, such as adding an appropriate layer to the base map when in dark mode, or a filter element to the pages containing events and helpful contacts. Lastly, and according to the potential mentioned in the Conclusion, the app could be adjusted to the business' target clientele. As a result of the easily customizable items, the establishments could modify the available events or predefined languages to better suit their intended customers.

7 Acknowledgements

I would like to thank my advisor, Remo Suppi, for his encouragement of my initial idea. Later on, his expertise and support were instrumental in the successful completion of the project.

I would also like to express my gratitude to everyone who helped by testing the PWA on their devices. This ensured that the software behaves correctly, even when faced with unforeseen variables outside of the controlled environment of emulation.

References

[1] Responsible Tourism. “Overtourism Solutions: Strategies to Manage Mass Tourism”. Available at: https://responsibletourismpartnership.org/overtourism-solutions/

[2] Rachel Brown. “Progressive Web Apps (PWAs): Definition, How They Work, and Why You Need One?”, 2019-02-22. Available at: https://ionic.io/resources/articles/what-is-a-progressive-web-app-and-why-you-need-one

[3] Ivan Metha. “Apple reverses decision about blocking web apps on iPhones in the EU”, 2024-03-01. Available at: https://techcrunch.com/2024/03/01/apple-reverses-decision-about-blocking-web-apps-on-iphones-in-the-eu/

[4] Iñaki de las Heras. “La economía española se hace cada vez más dependiente del turismo”, 2024-12-26. Available at: https://www.lavanguardia.com/economia/20241226/10238645/pib-turistico-supera-184-000-millones-euros-2023-record-fecha.html.

[5] María Jesús Ibáñez. “Estos son los municipios españoles que más masificación turística soportan”, 2024-08-08. Available at: https://www.elperiodico.com/es/economia/20240808/turismo-masificado-ranking-pueblos-saturados-106703853

[6] “Lloret Smart App”. Available at: https://apps.apple.com/es/app/lloret-smart/id1133201028?l=en-GB

[7] Danny Bluestone. “How to Combine User-Centered Design and Agile Development”. Available at: https://www.uxmatters.com/mt/archives/2015/12/how-to-combine-user-centered-design-and-agile-development.php

[8] “Ionic Docs”. Available at: https://ionicframework.com/docs.

[9] The project’s GitHub Repository. Available at: https://github.com/samozakod/PWA-GCIS.

[10] The Ionic Conference App’s GitHub Repository. Available at: https://github.com/ionic-team/ionic-conference-app.

[11] Figma, Inc. “Figma: The Collaborative Interface Design Tool”. Available at: https://www.figma.com/.

[12] W3Schools. “HTML ¡iframe¿ Tag”. Available at: https://www.w3schools.com/TAGS/tag_iframe.asp

[13] Meteorological Service of Catalonia. “Meteocat Weather Widgets”. Available at: https://static-m.meteo.cat/ginys/

[14] Allen Channing. “Twitter’s new login requirement will make embedded tweets a thing of the past”, 2023-07-02. Available at: https://www.indiehackers.com/post/twitters-new-login-requirement-will-make-embedded-tweets-a-thing-of-the-past-df825a42ec.

[15] Lloret Gaceta. Available at: https://lloretgaceta.com/

[16] What’s Up Cams. Available at: https://www.whatsupcams.com/en/

[17] Stack Overflow. “How to use Video.js with Ionic 2 + Angular 2”. Available at: https://stackoverflow.com/questions/47006698/how-to-use-video-js-with-ionic-2-angular-2

[18] Ionic Framework Forum. “Playing m3u8 on ionic 3”. Available at: https://forum.ionicframework.com/t/playing-m3u8-on-ionic-3/141074

[19] Leaflet. “Leaflet API Reference”. Available at: https://leafletjs.com/reference.html

[20] Per Liedman. “Leaflet Control Geocoder” Repository. Available at: https://github.com/perliedman/leaflet-control-geocoder

[21] Apache. “Awesome Cordova Plugins” Repository. Available at: https://github.com/apache/cordova-plugin-geolocation

[22] W3Schools. “JSON Introduction”. Awesome Cordova Plugins. Available at: https://www.w3schools.com/js/jsjsonintro.asp

[23] Thomas Pointhuber. “Leaflet Color Markers” Repository. Available at: https://github.com/pointhi/leaflet-color-markers

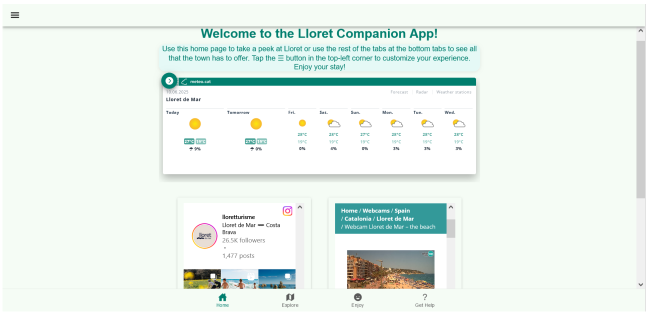
[24] Ngx-Translate. “ngx-translate” Repository. Available at: https://github.com/ngx-translate

[25] Stack Overflow. “How to deploy Ionic 4 app to Github pages?” Repository. Available at: https://stackoverflow.com/questions/53036381/how-to-deploy-ionic-4-app-to-github-pages

Appendix

A.1 Results Overview

Throughout *Figures 7 to 12*, the final appearance of the PWA’s main content is shown via screenshots, taken through a desktop browser.



Welcome to the Lloret Companion App!

Use this home page to take a peek at Lloret or use the rest of the tabs at the bottom tabs to see all that the town has to offer. Tap the ☰ button in the top-left corner to customize your experience. Enjoy your stay!

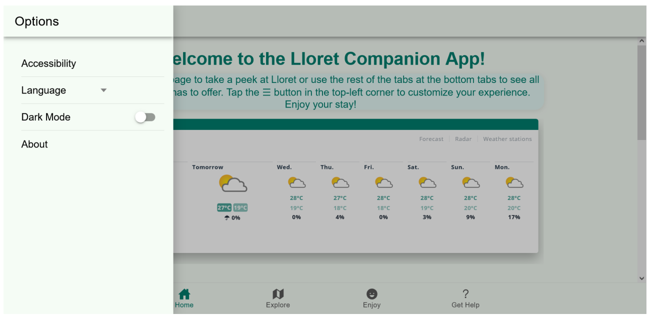
meteo.cat
10.06.2025
Lloret de Mar

Today	Tomorrow	Fri.	Sat.	Sun.	Mon.	Tue.	Wed.
27°C 19°C	27°C 19°C	28°C 19°C	28°C 19°C	27°C 19°C	28°C 19°C	28°C 19°C	28°C 19°C
9%	0%	0%	4%	0%	3%	3%	3%

lloretturisme
Lloret de Mar ➖ Costa Brava
26.5K followers
1,477 posts

Home / Webcams / Spain / Catalonia / Lloret de Mar / Webcam Lloret de Mar – the beach

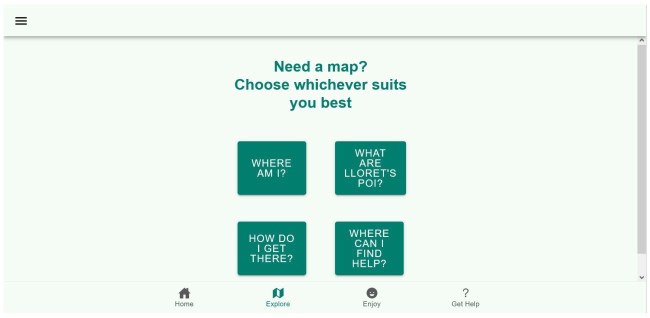
Fig. 7: Screenshot of the app’s ’Home’ page



elcome to the Lloret Companion App!

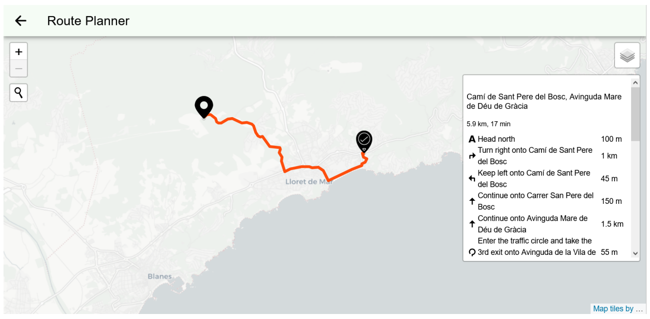
age to take a peek at Lloret or use the rest of the tabs at the bottom tabs to see all
has to offer. Tap the ☰ button in the top-left corner to customize your experience.
Enjoy your stay!

Fig. 8: Screenshot of the app's 'Options' side menu



Need a map?
Choose whichever suits you best

Fig. 9: A screenshot of the app's 'Explore' page

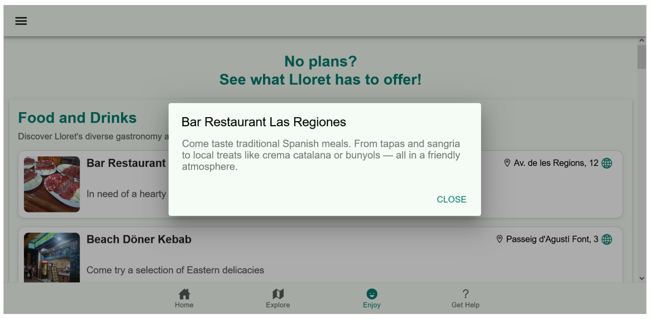


Camí de Sant Pere del Bosc, Avinguda Mare de Déu de Gràcia

5.9 km, 17 min

A	Head north	100 m
↱	Turn right onto Camí de Sant Pere del Bosc	1 km
↰	Keep left onto Camí de Sant Pere del Bosc	45 m
↑	Continue onto Carrer San Pere del Bosc	150 m
↑	Continue onto Avinguda Mare de Déu de Gràcia	1.5 km
⟲	Enter the traffic circle and take the 3rd exit onto Avinguda de la Vila de	55 m

Fig. 10: Screenshot of one of the 'Explore' section's maps



No plans?
See what Lloret has to offer!

Food and Drinks

Discover Lloret's diverse gastronomy a

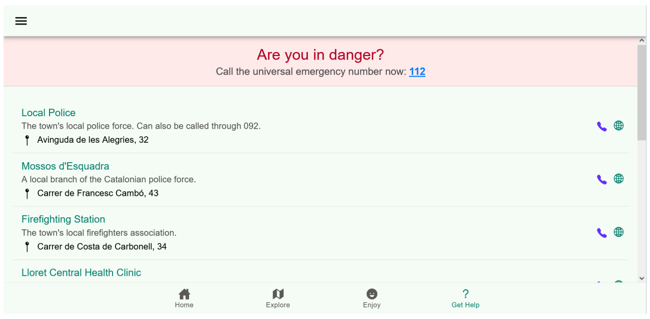
Bar Restaurant
In need of a hearty
Av. de les Regions, 12

Bar Restaurant Las Regiones

Come taste traditional Spanish meals. From tapas and sangria to local treats like crema catalana or bunyols — all in a friendly atmosphere.

Beach Döner Kebab
Passeig d'Agustí Font, 3
Come try a selection of Eastern delicacies

Fig. 11: Screenshot of the app's 'Enjoy' page



Are you in danger?
Call the universal emergency number now: 112

Local Police
The town's local police force. Can also be called through 092.
Avinguda de les Alegries, 32

Mossos d'Esquadra
A local branch of the Catalonian police force.
Carrer de Francesc Cambó, 43

Firefighting Station
The town's local firefighters association.
Carrer de Costa de Carbonell, 34

Lloret Central Health Clinic

Fig. 12: Screenshot of the app's 'Get Help' page