PORTING THE INDICO CHECK-IN TO A PROGRESSIVE WEB APP

August 2023

AUTHOR:
João Mesquita

FEUP, Informatics and Computer Engineer

SUPERVISORS:
Tomas Roun, CERN
Adrian Monnich, CERN

ABSTRACT

In the dynamic landscape of event management, the transition from conventional systems to modern and versatile solutions is imperative. This report delves into the transformation of the Indico Check-In application, migrating it from a legacy AngularJS-based mobile app to a state-of-the-art Progressive Web App (PWA).

The paper begins by introducing the aim of the project, followed by an exploration of Progressive Web Apps, highlighting their advantages in providing native-like experiences, advanced capabilities, web-related benefits, and development costs. Conversely, the report addresses the challenges and trade-offs associated with PWAs. The existing Indico Check-In application is explained to discern the limitations of the previous system, setting the stage for the evolution to a PWA.

The report then moves to the conceptualization and design section, where all the efforts before moving into the development phase are explained and why they were important. From requirements extraction to the investigation of different tech stacks, this phase is essential to avoid committing to a bad decision. Moreover, a comparison between the previous and the new application is shown for different pages.

The implementation phase witnesses the strategic utilization of React, Tailwind CSS, and linter tools like ESLint and Prettier to uphold coding standards and streamline development. An outline of the development roadmap is presented, as well as an explanation of the adopted coding practices and examples. Lastly, the results of a performance analysis test using Lighthouse are presented and explained.

The report concludes by outlining the roadmap for future enhancements, emphasizing user experience upgrades, increasing the test coverage, and the addition of new features supported on the Indico Website. Finally, a summary of the work is presented, describing the success of the project and the next steps for the Indico Check-In App.

KEYWORDS

cern, openlab, progressive web app, pwa, react, mobile development, application, indico, indico check-in app, porting, check-in, event management, performance, tailwind css, qr-code reader

TABLE OF CONTENTS

1 INTRODUCTION

In the ever-evolving landscape of mobile applications development, the shift towards Progressive Web Apps (PWAs) has become increasingly prevalent. This report describes the process of porting the Indico Check-In App from AngularJS to this new technology. The aim was to replace the aging AngularJS-based application with a cutting-edge React [11] PWA, that offers better performance, maintainability, and extensibility.

The previous Indico Check-In App is a tool to manage complex conferences, workshops, and meetings. It is used by event managers to control access to events. Therefore, the system provides a subset of the features offered by the Indico web application [6], to support on-site event management.

However, as technological advancements and user expectations evolved, it became imperative to revisit the architecture and capabilities of the system to ensure it remains relevant and effective at performing its task. This paper details the development process and outcomes of porting the app to a PWA, designed to replace the legacy system, while also introducing new features, a more intuitive interface, and cross-platform compatibility.

Throughout this report, we will delve into the key stages of the development process, from conceptualization and design to implementation and future work. Additionally, we will highlight the innovative features and improvements that the new PWA brings to the table, showcasing its potential to elevate the event management experience for both organizers and developers.

The homepage of the new App can be seen in Figure 1.

1.1 Code Repository

The project is open-source and accessible at `https://github.com/indico/indico-checkin-pwa`.

2 Progressive Web Apps

PWAs are a fairly recent technology that has been gaining popularity over the years. A progressive web app is an application built using web technologies that can be installed and run on all devices, from one codebase. It provides a native-like experience on supporting devices, while also being capable of running in the browser. Hence, the benefits of this technology are evident. It leverages the reach of Web Applications, whilst offering the capabilities of Native Apps, as depicted in Figure 2.

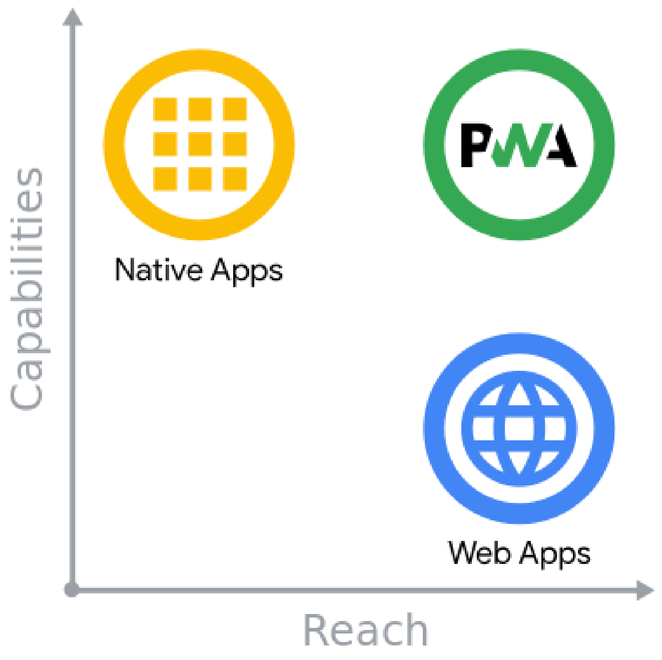


Figure 2: capabilities vs. reach of Native Apps, Web Apps and PWAs [14]

2.1 Advantages of PWAs

Adopting this technology is a great way to make an app safe, discoverable, linkable, easy to install and update, responsive, and network independent. There are many success stories of companies that converted their applications to PWAs. *Starbucks*, for example, has increased daily active users twice in their app. Another example is *Tinder* which was able to reduce loading times by a factor of 3 with their PWA. The resulting app was also 90% smaller than the compiled Android app [12].

2.1.1 Native-like experiences

When installed on a device, PWAs function just like other native apps and are viewed by the user as a regular app. For example:

- Application icons that can be added to a device's home screen or taskbar.
- Launch automatically when an associated file type is opened.
- Download from application stores, such as the Play Store or Microsoft Store.

2.1.2 Advanced Capabilities

PWAs also have access to advanced capabilities. For example:

- Function in offline mode and support push notifications.
- Perform periodic updates even when the application is not running.
- Access some hardware features, such as the camera and GPS.

2.1.3 Web-Related Advantages

PWAs can run in web browsers, just like websites, giving them advantages such as:

- Indexed by search engines and shareable via a web link.
- Safe for users because they use secure HTTPS endpoints and other user safeguards.
- Adaptive to the user's screen size or orientation, and input method.

2.1.4 Lower Development Cost

PWAs have a much lower cross-platform development cost than compiled apps that require a separate codebase for each platform, such as Android, iOS, and each desktop operating system. With a PWA, you can use a single codebase that's shared between your website, mobile app, and desktop app.

Progressive Web Apps achieve this by utilizing the underlying browser infrastructure. A simplified way to conceptualize PWAs is as web applications that conceal the browser interface, delivering users an experience akin to using a traditional mobile app.

2.2 Disadvantages of PWAs

While there are several advantages to using progressive apps, there are some drawbacks that need to be accounted for [17]:

- Compatibility with iOS: Apple doesn't allow PWAs to access many important features, including Touch ID, Face ID, ARKit, Bluetooth, serial, Beacons, altimeter sensor, and even battery information.
- Issues with Legacy Devices: Older mobile devices with outdated web browsers don't work flawlessly. While this problem will inevitably solve itself in the future, it may be a hassle during the transition period.
- Capability Constraints: Progressive apps can't do everything mobile apps can do. Because they are written in JavaScript, they are not as efficient as apps written in native languages, such as Kotlin or Swift. Moreover, they have limited access to hardware components.

3 Previous Indico Check-In App

Before elaborating on this project's work, it is necessary to give some context about the current Indico Check-In App. It consists of a Check-In application for Indico events that uses a QR code system to add new events and check-in participants to these events. In addition to this, the app allows the visualization of the event's and participants' details. Other features like participant search and configuration settings to perform auto check-in are also present.

The sitemap of the mobile app can be seen in Figure 3. You may notice the existence of a *Registration Form Page* that has not been mentioned before. That is because the sitemap belongs to the new PWA and the addition of Registration Forms was part of this work. Therefore, each Indico event may contain several Registration Forms. A Registration Form is equivalent to a sub-event, take as an example a networking session inside a conference. Participants may attend the conference talks without attending the networking session, thus these 2 sub-events are separated and each corresponds to a registration form. Finally, a Registration Form may contain multiple participants and there is also a dedicated page to display the participant's data necessary for the event.

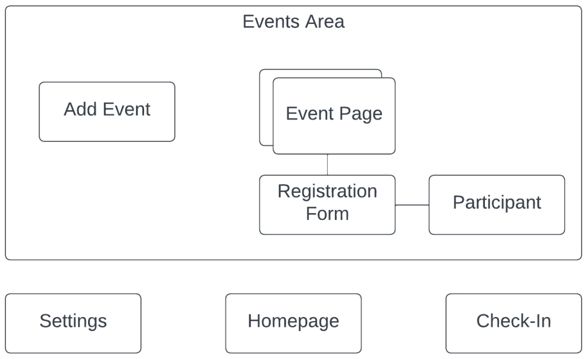


Figure 3: Indico Check-In App Sitemap

4 Conceptualization and Design

The transition from the legacy AngularJS application to a Progressive Web App required a comprehensive analysis of the system's architecture and design.

4.1 Conceptualization

The first step of the conceptualization phase was to assess the requirements for the project. This step was facilitated by the existence of the previous check-in app, which contains the main features to implement in the new system. Nonetheless, it was crucial to outline key points of improvement right from the start. These were:

1. Increase the QR code reading speed.
2. Modernize the UI and improve the User Experience.
3. Create a maintainable and upgradeable React codebase.
4. Provide additional information regarding the events.
5. Separate Events from Registration Forms.

From here, the next step was to choose the tech stack that would be used to develop the new check-in app. Actually, the initial project idea was to port the current application to a React Native application. It was only decided to move on with a PWA after weighing down the good and bad of each alternative. Hence, the 3 alternatives were:

1. React Native Application
2. Expo
3. Progressive Web App

Introducing the alternatives, React Native [10], an extension of the React library developed by *Meta*, enables the creation of mobile applications with a single codebase. Inspired by React's component-based architecture, React Native bridges the gap between native and web development, allowing developers to build high-quality mobile applications for multiple platforms simultaneously. Its ability to deliver native-like performance and user experiences on both iOS and Android has made it a popular choice for cross-platform development.

Meanwhile, Expo [3], a platform built around React Native, facilitates the development of cross-platform applications. By providing a unified environment and a rich set of pre-built components, Expo expedites the creation of applications for both iOS and Android platforms. This accelerates the development cycle, reduces platform-specific complexities, and promotes consistent user experiences across devices.

The decision was influenced by all the advantages explained above of PWAs. However, another important factor was to test the QR Code reader using each of the technologies. For this reason, a proof-of-concept application [9] was developed using each of the 3 to verify the capabilities and flexibility of the QR readers. Using React with the HTML5-QR-reader [15] JavaScript library was deemed the better option due to its customization and documentation.

To complement the use of React, Tailwind CSS [16] was used for styling the application, since it allows the creation of a unique design system according to the user needs. This utility may introduce some overhead in the development process in the beginning, but through the creation of reusable components, the final outcome is a custom-made UI that stands out from the norm.

At this point, after having the requirements laid out and the technology stack settled, the only necessary step missing is the database structure, since the system stores events, registration forms, participants, and server data in the device. IndexedDB [7] was utilized to perform this task. It is a JavaScript-based object-oriented database that offers client-side storage of significant amounts of structured data. This database uses indexes to enable high-performance searches of this data.

The structure of the offline database can be viewed in Figure 4.

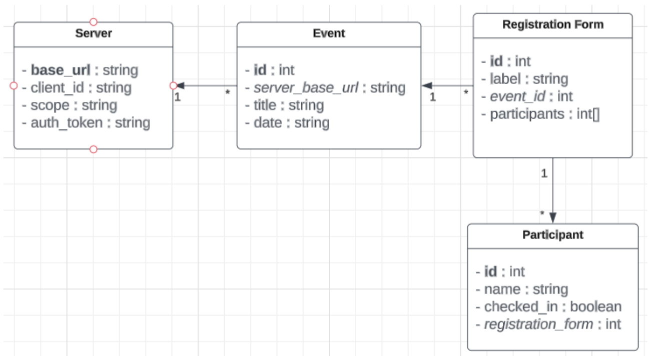
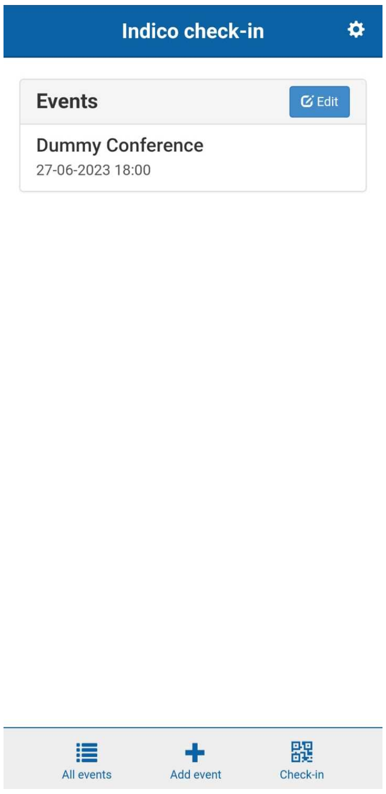


Figure 4: Database UML Classes Diagram

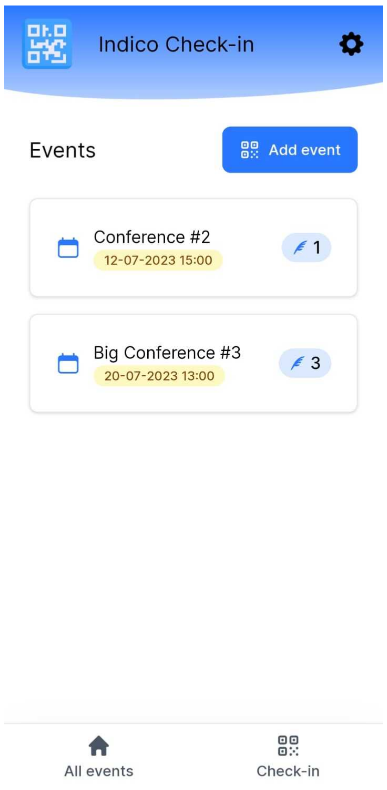
4.2 Design

When building an app from scratch, the user interface and user experience occupy a significant amount of time for good reasons. Having a solid design foundation and mock-ups speeds up the development process and avoids unnecessary refactors in the future. In the context of this project, one of the main objectives was to redesign the UI and create an intuitive, minimalistic, and modernized application.

The main challenge here was to abstract from the existing AngularJS app and come up with something fresh that would not resemble the previous one. To accomplish this, several design ideas were analyzed and tested using Figma [4]. Consequently, the PWA app is easily distinguished from the previous one, as confirmed by Figure 5 and Figure 6. Furthermore, another great addition to the new app is the possibility to switch between white and dark modes.



(a) AngularJS Homepage



(b) PWA Homepage

Figure 5: Old vs. New Homepage in White mode

5 Implementation

The development process was the longest phase of the project. It required frequent communication with Indico's backend team to coordinate the API requests and create new routes for the new features. An overview of the development roadmap is present in Figure 7. Evidently, the mentioned graphic is a very simplified representation of the work done, where each box groups several sub-tasks.

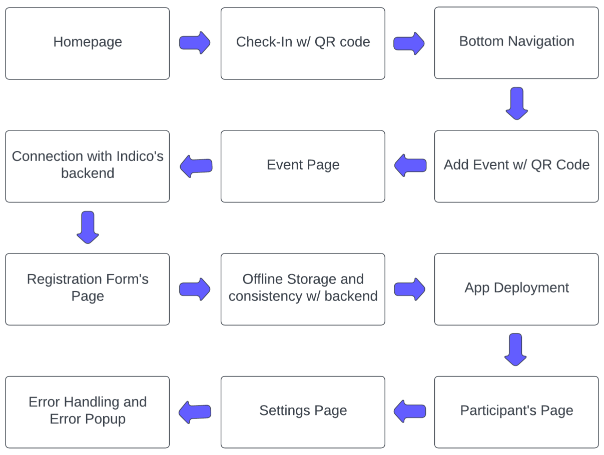


Figure 7: Development Roadmap

5.1 Development Environment

This PWA was built using React and Tailwind CSS for styling. The app supports both TypeScript and JavaScript files to take advantage of type safety, along with other benefits of the former and the flexibility and speed-up of the latter. Thus, components that contain an interface to the outside, such as design components or state objects, are organized into TypeScript files.

The code was developed on *Visual Studio Code*, with the aid of code versioning tools: *git* and *GitHub*. *Mattermost* was used for team communication and there were weekly review meetings either on-site or via *Zoom*, where we detailed the tasks performed during the last week and planned the work for the current one.

5.2 Coding Practices and Examples

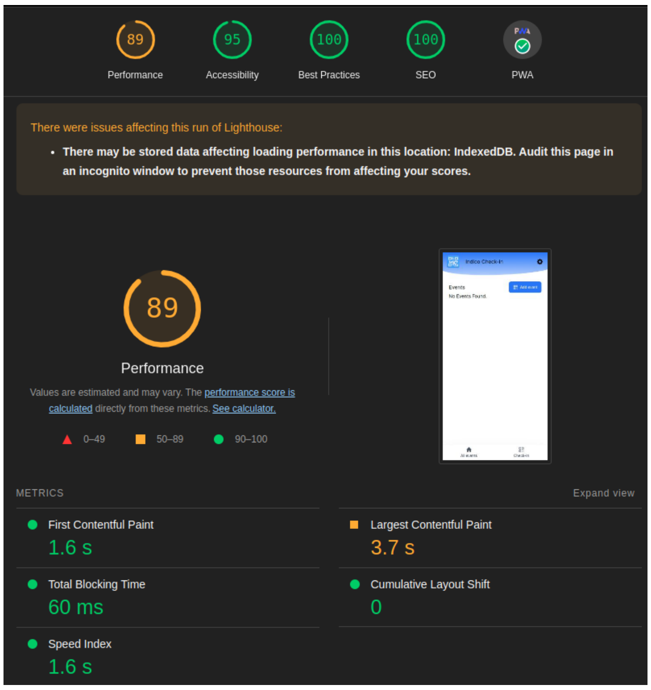
Maintaining code consistency and quality is paramount to the success of any software project when working in a collaborative environment. These qualities are maintained through various methods, one of them being code reviews. However, it is a time-consuming process and it's possible to optimize it. For this reason, a static code analyzer, ESLint [2], is used along with Prettier [13] to ensure that good programming practices are applied to the code. Not only that, it also preserves formatting consistency in the codebase.

Moreover, since the project only has 2 contributors as of the time of writing, it was simple to manage the git repository. Anyhow, some practices from the GitFlow Workflow [1] were adopted, such as the naming conventions for branches and commit messages. Whenever a branch was ready to be merged, a pull request was created to undergo review before inserting the changes in the master branch.

Lastly, some code examples demonstrating the coding style adopted in the project are shown in Figure 8.

5.3 Audit and Performance Analysis

In the pursuit of delivering a high-performance and user-friendly Progressive Web App, an audit and performance analysis were performed to evaluate the application's speed, coding practices, accessibility, SEO (search engine optimization), and PWA compatibility. The Lighthouse Report [5], a comprehensive testing tool provided by Google, was employed to assess these critical aspects and identify areas for improvement. The results shown in Figure 9 were very positive while, at the same time, foreshadowing one of the main tasks considered for future work, which is the caching of network requests, that will improve the Performance indicator.



There were issues affecting this run of Lighthouse:

- **There may be stored data affecting loading performance in this location: IndexedDB. Audit this page in an incognito window to prevent those resources from affecting your scores.**

89

Performance

Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

▲ 0–49 ■ 50–89 ● 90–100

METRICS Expand view

● First Contentful Paint 1.6 s	■ Largest Contentful Paint 3.7 s
● Total Blocking Time 60 ms	● Cumulative Layout Shift 0
● Speed Index 1.6 s	

Figure 9: Lighthouse App Report

6 Future Work and Conclusions

6.1 Future Work

As the minimum-viable product is concluded, the path ahead for this system is filled with possibilities for growth and innovation. Some of the future improvements are:

1. Reduce the loading times of the application by caching network requests and updating the app at a given interval of time.

2. Introduce offline mode to the app. Currently, the app already stores the data in a local DB, so the offline mode could provide limited functionality while the app cannot connect to the backend.

3. Add section details to the Participant's page. Indico events may contain flexible data fields for each participant in a Registration Form. An initial effort to introduce these flexible fields into the Participant's page is already present in a specific branch on the official GitHub.

4. Improve test coverage. Introduce unit tests and usability tests to collect feedback from users.

5. Handle authentication token expiration. In case the app's authentication token to an Indico server expires, it could attempt to automatically refresh it for a seamless experience.

6. Cross-Platform compatibility. Ensure that the app functions as expected on older systems and browsers.

6.2 Conclusion

In the realm of Indico and event management, where efficiency and ease of use are paramount, the transformation from a legacy AngularJS-based system to a state-of-the-art React Progressive Web App stands as an important milestone that will contribute to better event management and user experience. This report encapsulates the journey of conceptualization, design, implementation, and optimization, setting the stage for a modern and seamless event management experience.

The decision to embrace PWAs was rooted in the vision to elevate reach, user satisfaction, and maintainability. Through meticulous planning, the new app introduces event organizers to a modern interface to check in attendees, add events, and view current events, registration forms, and even participant data regarding each registration form. On top of that, multiple features to improve the user experience were included, such as a dark theme and swiping gestures to switch between the bottom tabs.

Built upon the React framework and the flexible styling of Tailwind CSS, the development environment embodies adaptability and efficiency. The codebase was designed to facilitate maintenance and the introduction of new functionalities. After all, the Indico Check-In App is an open-source project that is open to contributions from anyone, making readability and code standards crucial properties.

To conclude, the migration to a Progressive Web App was a success. The features present in the previous application have been ported to the new system. Moreover, the PWA offers a modern design that is intuitive and passes the Lighthouse accessibility test with a great score. The new application extended the feature set, improving the experience of event managers and opening the way for future development in a maintainable environment that uses a technology stack familiar to the Indico team.

7 REFERENCES

[1] Atlassian Bitbucket. *Gitflow Workflow*. URL: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow (visited on 08/08/2023).

[2] ESLint. *ESLint website*. URL: https://eslint.org/ (visited on 08/08/2023).

[3] Expo. *Expo website*. URL: https://expo.dev/ (visited on 08/09/2023).

[4] Figma. *Figma getting started*. URL: https://help.figma.com/hc/en-us/categories/360002051613 (visited on 08/07/2023).

[5] Google. *Lighthouse Report Website*. URL: https://developer.chrome.com/docs/lighthouse/overview/ (visited on 08/10/2023).

[6] Indico. *Indico Website*. URL: https://indico.cern.ch/ (visited on 08/09/2023).

[7] mdn. *IndexedDB documentation*. URL: https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API (visited on 08/09/2023).

[8] João Mesquita. *Indico Check-In PWA App Deployment*. URL: https://indico-check-in-app.onrender.com/ (visited on 08/09/2023).

[9] João Mesquita. *QRCode Reader proof-of-concept Apps*. URL: https://github.com/monkin77/qr-reader-apps (visited on 08/07/2023).

[10] Meta. *React Native website*. URL: https://reactnative.dev/ (visited on 08/09/2023).

[11] Meta. *React website*. URL: https://react.dev/ (visited on 08/09/2023).

[12] MSEdgeTeam. *Overview of Progressive Web Apps (PWAs)*. URL: https://learn.microsoft.com/en-us/microsoft-edge/progressive-web-apps-chromium/ (visited on 08/09/2023).

[13] Prettier. *Prettier website*. URL: https://prettier.io/ (visited on 08/08/2023).

[14] Sam Richard and Pete LePage. *What are Progressive Web Apps?* URL: https://web.dev/what-are-pwas/ (visited on 08/09/2023).

[15] ScanApp. *Html5-Qrcode JS Library*. URL: https://scanapp.org/html5-qrcode-docs/ (visited on 08/07/2023).

[16] Tailwindcss. *Tailwindcss utility*. URL: https://tailwindcss.com/ (visited on 08/08/2023).

[17] Matt Warcholinski. *Advantages and Disadvantages of Progressive Web Apps*. URL: https://brainhub.eu/library/progressive-web-apps-advantages-disadvantages (visited on 08/09/2023).

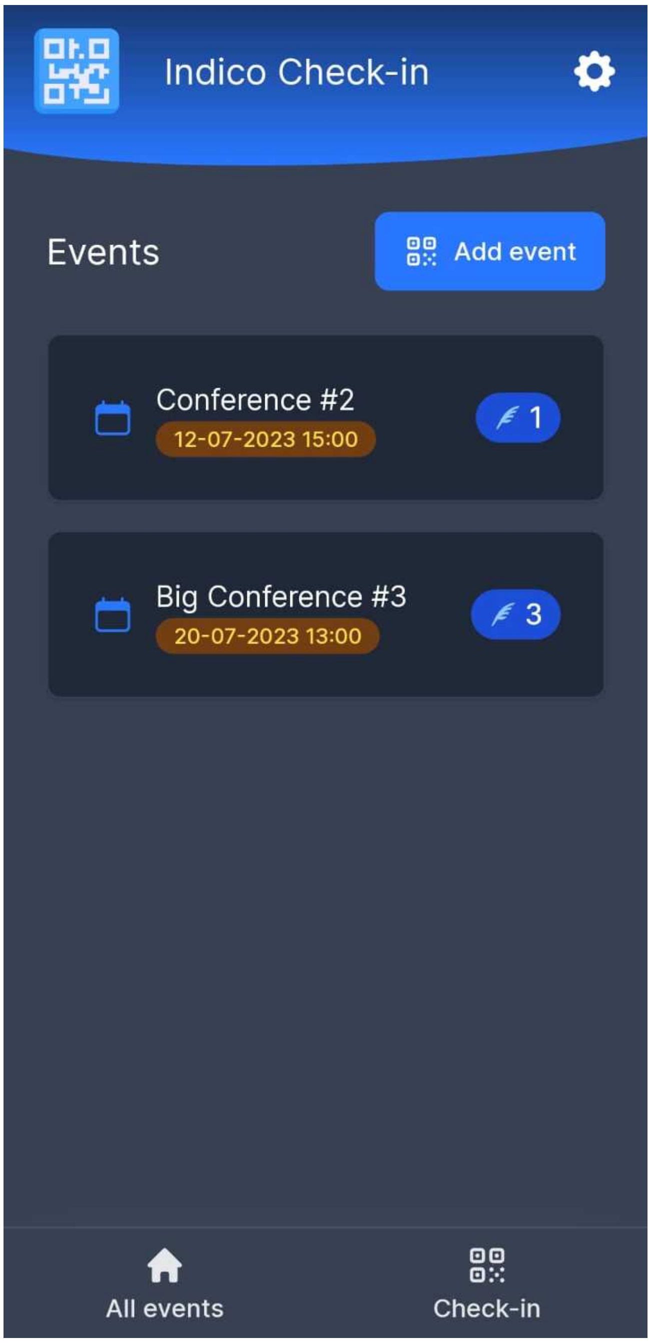
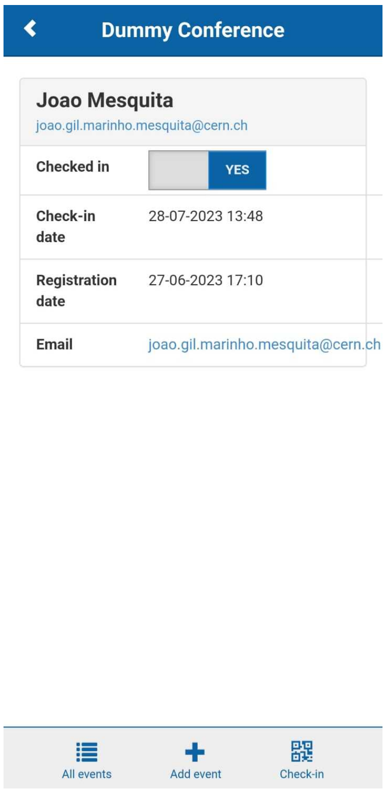


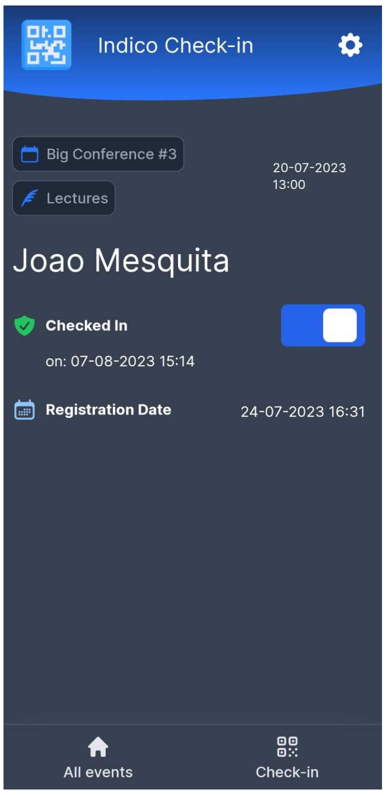
Figure 1: New Indico Check-In App Homepage [8]


Dummy Conference

Joao Mesquita
joao.gil.marinho.mesquita@cern.ch

Checked in	YES
Check-in date	28-07-2023 13:48
Registration date	27-06-2023 17:10
Email	joao.gil.marinho.mesquita@cern.ch

(a) AngularJS Participant's page


Indico Check-in

Big Conference #3
Lectures
20-07-2023 13:00

Joao Mesquita

Checked In
on: 07-08-2023 15:14

Registration Date 24-07-2023 16:31

(b) PWA Participant's page

Figure 6: Old vs. New Participant's page

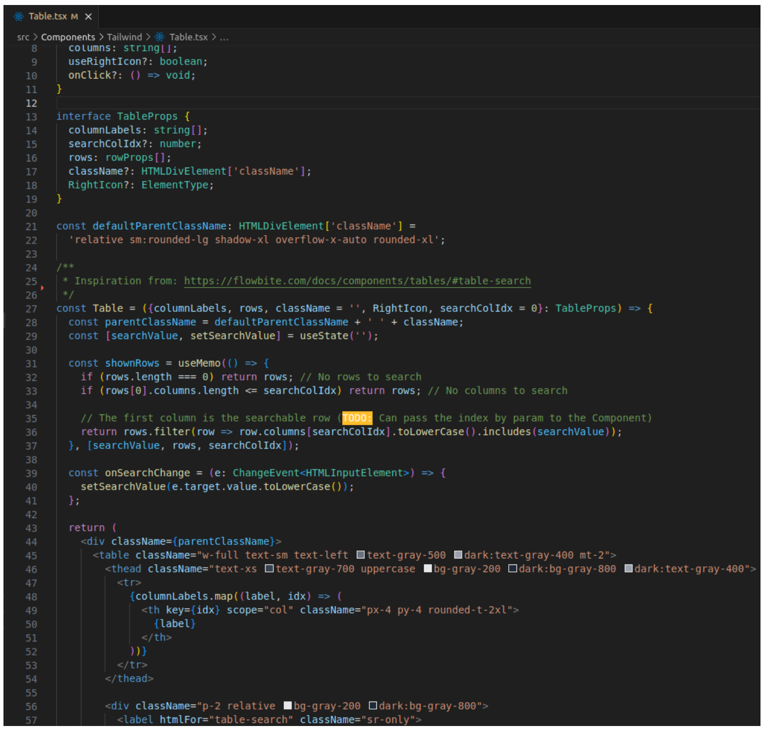
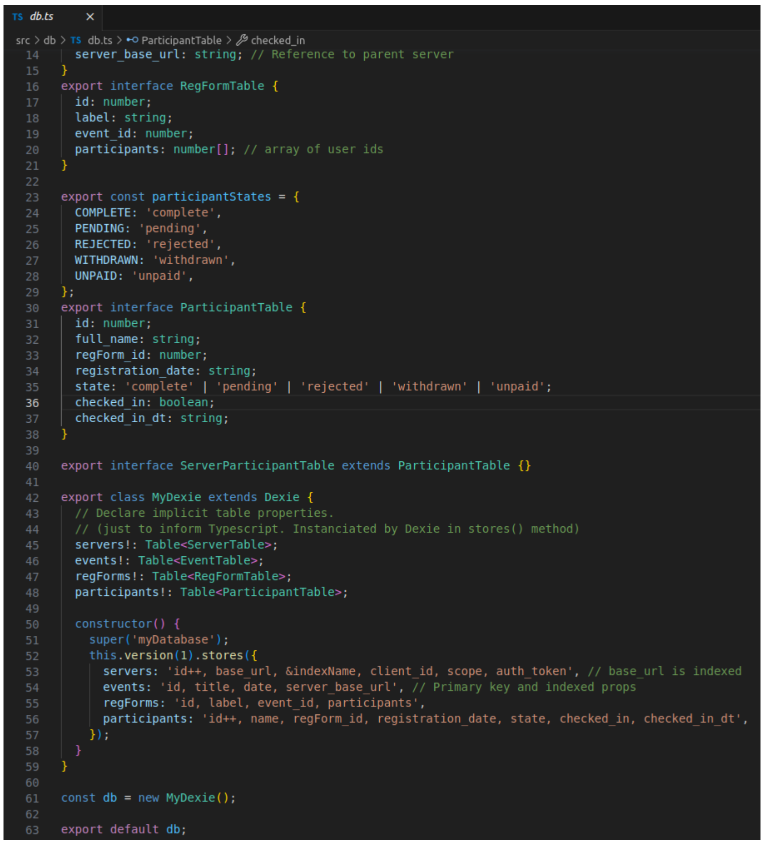


Table.tsx M ×

src > Components > Tailwind > Table.tsx > ...

```tsx
  columns: string[];
  useRightIcon?: boolean;
  onClick?: () => void;
}

interface TableProps {
  columnLabels: string[];
  searchColIdx?: number;
  rows: rowProps[];
  className?: HTMLDivElement['className'];
  RightIcon?: ElementType;
}

const defaultParentClassName: HTMLDivElement['className'] =
  'relative sm:rounded-lg shadow-xl overflow-x-auto rounded-xl';

/**
 * Inspiration from: https://flowbite.com/docs/components/tables/#table-search
 */
const Table = ({columnLabels, rows, className = '', RightIcon, searchColIdx = 0}: TableProps) => {
  const parentClassName = defaultParentClassName + ' ' + className;
  const [searchValue, setSearchValue] = useState('');

  const shownRows = useMemo(() => {
    if (rows.length === 0) return rows; // No rows to search
    if (rows[0].columns.length <= searchColIdx) return rows; // No columns to search

    // The first column is the searchable row (TODO: Can pass the index by param to the Component)
    return rows.filter(row => row.columns[searchColIdx].toLowerCase().includes(searchValue));
  }, [searchValue, rows, searchColIdx]);

  const onSearchChange = (e: ChangeEvent<HTMLInputElement>) => {
    setSearchValue(e.target.value.toLowerCase());
  };

  return (
    <div className={parentClassName}>
      <table className="w-full text-sm text-left text-gray-500 dark:text-gray-400 mt-2">
        <thead className="text-xs text-gray-700 uppercase bg-gray-200 dark:bg-gray-800 dark:text-gray-400">
          <tr>
            {columnLabels.map((label, idx) => (
              <th key={idx} scope="col" className="px-4 py-4 rounded-t-2xl">
                {label}
              </th>
            ))}
          </tr>
        </thead>

        <div className="p-2 relative bg-gray-200 dark:bg-gray-800">
          <label htmlFor="table-search" className="sr-only">
```

(a) Custom Tailwind CSS Table Component



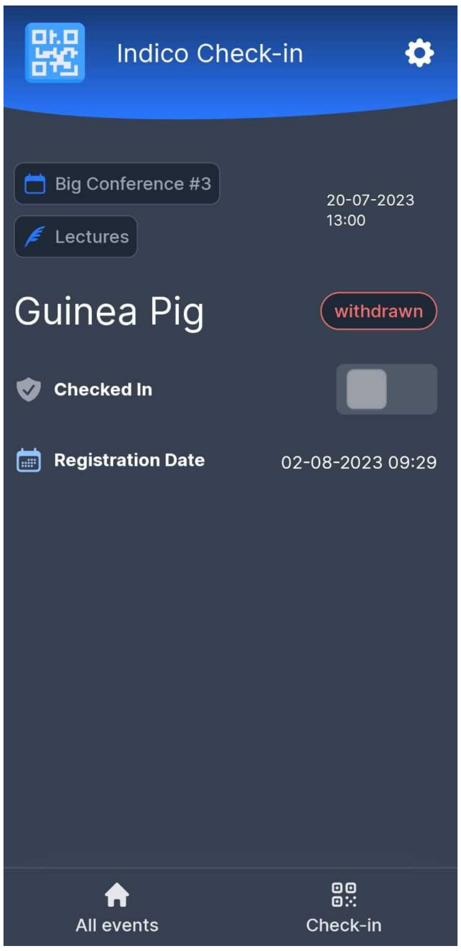
TS db.ts ×

src > db > TS db.ts > ParticipantTable > checked_in

```ts
  server_base_url: string; // Reference to parent server
}
export interface RegFormTable {
  id: number;
  label: string;
  event_id: number;
  participants: number[]; // array of user ids
}

export const participantStates = {
  COMPLETE: 'complete',
  PENDING: 'pending',
  REJECTED: 'rejected',
  WITHDRAWN: 'withdrawn',
  UNPAID: 'unpaid',
};
export interface ParticipantTable {
  id: number;
  full_name: string;
  regForm_id: number;
  registration_date: string;
  state: 'complete' | 'pending' | 'rejected' | 'withdrawn' | 'unpaid';
  checked_in: boolean;
  checked_in_dt: string;
}

export interface ServerParticipantTable extends ParticipantTable {}

export class MyDexie extends Dexie {
  // Declare implicit table properties.
  // (just to inform Typescript. Instanciated by Dexie in stores() method)
  servers!: Table<ServerTable>;
  events!: Table<EventTable>;
  regForms!: Table<RegFormTable>;
  participants!: Table<ParticipantTable>;

  constructor() {
    super('myDatabase');
    this.version(1).stores({
      servers: 'id++, base_url, &indexName, client_id, scope, auth_token', // base_url is indexed
      events: 'id, title, date, server_base_url', // Primary key and indexed props
      regForms: 'id, label, event_id, participants',
      participants: 'id++, name, regForm_id, registration_date, state, checked_in, checked_in_dt',
    });
  }
}

const db = new MyDexie();

export default db;
```

(b) Database definition

Figure 8: Code examples


Indico Check-in

Big Conference #3

Lectures

20-07-2023 13:00

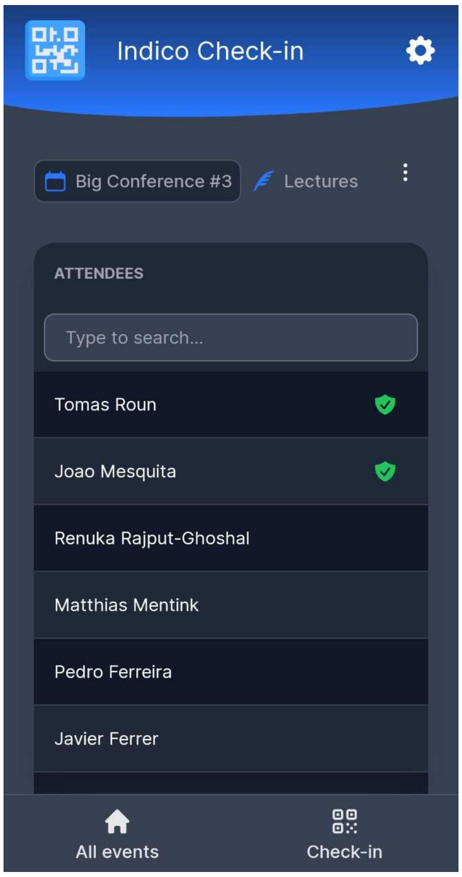
Guinea Pig

withdrawn

Checked In

Registration Date 02-08-2023 09:29

(a) New Participant's Page on rejected status


Indico Check-in

Big Conference #3

Lectures

ATTENDEES

Type to search...

Tomas Roun

Joao Mesquita

Renuka Rajput-Ghoshal

Matthias Mentink

Pedro Ferreira

Javier Ferrer

(b) New Registration Form Page

Figure 10: New App UI examles

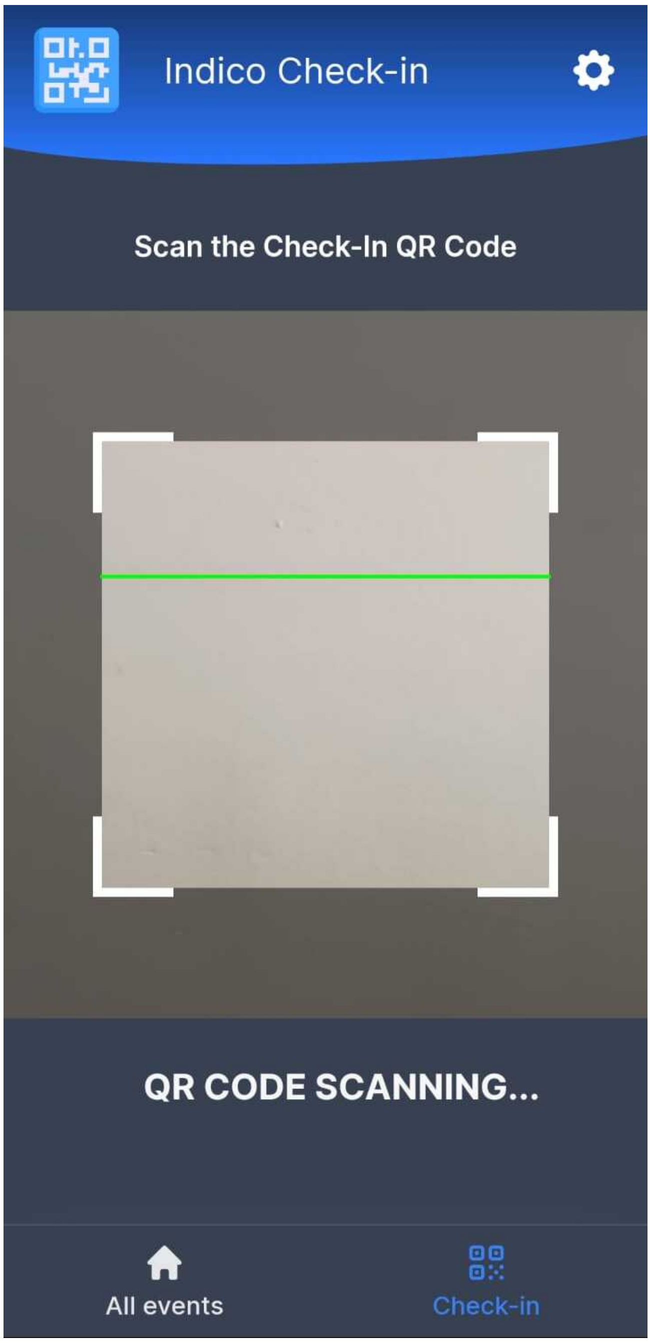


Figure 11: New App QR code Reader