

SKRIPSI

PEMBUATAN ULANG APLIKASI WSDC (*WORLD SCHOOL DEBATING CHAMPIONSHIP*) 2017 BALI DENGAN IONIC 5



Rajasa Cikal Maulana Solihin

NPM: 2017730084

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2022**

UNDERGRADUATE THESIS

**RE-CREATION OF WSDC (WORLD SCHOOL DEBATING
CHAMPIONSHIP) 2017 BALI APP WITH IONIC 5**



Rajasa Cikal Maulana Solihin

NPM: 2017730084

**DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY
2022**

LEMBAR PENGESAHAN

PEMBUATAN ULANG APLIKASI WSDC (*WORLD SCHOOL DEBATING CHAMPIONSHIP*) 2017 BALI DENGAN IONIC 5

Rajasa Cikal Maulana Solihin

NPM: 2017730084

Bandung, 01 November 2022

Menyetujui,

Pembimbing

Pascal Alfadian, Nugroho, M.Comp.

Ketua Tim Penguji

Anggota Tim Penguji

Raymond Chandra Putra, M.T.

Mengetahui,

Ketua Program Studi

Mariskha Tri Adithia, P.D.Eng

PERNYATAAN

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa skripsi dengan judul:

PEMBUATAN ULANG APLIKASI WSDC (*WORLD SCHOOL DEBATING CHAMPIONSHIP*) 2017 BALI DENGAN IONIC 5

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhkan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,
Tanggal 01 November 2022

Rajasa Cikal Maulana Solihin
NPM: 2017730084

ABSTRAK

World Schools Debating Championships (WSDC) merupakan sebuah turnamen debat Bahasa Inggris tahunan untuk tim-tim tingkat sekolah menengah yang mewakili berbagai negara. Pada tahun 2017, WSDC diselenggarakan di Bali, Indonesia. Untuk menunjang acara tersebut, diciptakan sebuah aplikasi WSDC 2017 Bali yang memiliki beberapa fitur seperti melihat jadwal acara, melihat pengumuman acara, melihat lokasi dan peta lokasi acara, dan fitur notifikasi. Aplikasi tersebut dibangun menggunakan Ionic Framework versi 3 dengan Cordova dan diimplementasikan menggunakan *framework* JavaScript Angular serta dapat digunakan pada perangkat mobile berbasis Android dan iOS. Karena pada saat skripsi ini dibuat, Ionic Framework versi 3 sudah tidak lagi dikembangkan, dan aplikasi WSDC 2017 Bali sudah diturunkan dari App Store pada perangkat iOS karena tidak mendapat pembaruan sejak lama, maka dari itu dibuatlah aplikasi pembaruan aplikasi WSDC 2017 Bali menggunakan Ionic Framework versi 6.

Aplikasi WSDC 2017 Bali dengan Ionic Framework 6 diimplementasikan dengan menggunakan *framework* JavaScript Angular. Ionic Framework memiliki UI Component, yaitu berupa tag khusus yang digunakan untuk menambah fungsionalitas aplikasi, contohnya tag <ion-button> digunakan untuk menambah tombol. Ionic Framework dapat menggunakan fitur *native* dari perangkat dengan menggunakan Native Api, yaitu Capacitor. Capacitor memiliki *plugin* yang digunakan untuk mengakses fitur *native* pada perangkat mobile, seperti *Geolocation* dan *In App Browser*. *Plugin* yang digunakan pada aplikasi ini yaitu: Geolocation API, Google Maps API, Splash Screen, dan Browser API. Dengan digunakannya Capacitor maka aplikasi WSDC 2017 Bali yang ditulis menggunakan bahasa pemrograman web seperti HTML, CSS, dan JavaScript, dapat dijalankan pada perangkat *mobile* dengan sistem operasi Android dan iOS. Namun pada skripsi ini hanya akan diuji pada perangkat Android.

Pengujian aplikasi WSDC 2017 Bali dilakukan terhadap beberapa responden dengan cara mengunduh, menginstal, dan membandingkan aplikasi WSDC 2017 Bali terdahulu dan terbaru. Dari pengujian tersebut didapatkan hasil yaitu aplikasi dapat berjalan dengan lancar pada perangkat Android mulai dari versi 5.1 sampai dengan versi 11. Dengan begitu, aplikasi WSDC 2017 Bali dengan Ionic Framework versi 6 telah berhasil dibangun dan dijalankan pada perangkat Android versi 5.1 sampai dengan versi 11.

Kata-kata kunci: WSDC, Ionic Framework, UI Component, Capacitor, Cordova, Angular

ABSTRACT

The World Schools Debating Championships (WSDC) is an annual English language debate tournament for high school-level teams representing various countries. In 2017, WSDC was held in Bali, Indonesia. To support the event, a WSDC 2017 Bali application was created which has several features such as viewing the event schedule, displaying announcements, viewing the location and map of the event location, and notification features. The application was built using the Ionic Framework version 3 with Cordova and implemented using the Angular JavaScript framework. At the time this thesis was written, the Ionic Framework version 3 was no longer developed, and the Ionic Framework had reached version 6. Therefore, an update for the WSDC 2017 Bali application will be made using the Ionic Framework version 6.

The WSDC 2017 Bali application with Ionic Framework 6 is implemented using the Angular JavaScript framework. The Ionic Framework has a UI Component, which is a special tag that is used to add application functionality, for example the `<ion-button>` tag is used to add buttons. Ionic Framework can use the native features of the device by using the Native API, i.e. Capacitor. Capacitor has plugins that are used to access native features on mobile devices, such as Geolocation and In App Browser. The plugins used in this application are: Geolocation API, Google Maps API, Splash Screen, and Browser API. By using Capacitor, the WSDC 2017 Bali application, which is written using web programming languages such as HTML, CSS, and JavaScript, can be run on mobile devices with Android and iOS operating systems. However, this thesis will only be tested on Android devices.

Testing of the WSDC 2017 Bali application was carried out on several respondents by downloading, installing, and comparing the previous and latest WSDC 2017 Bali applications. From these tests, the results are that the application can run well on Android devices starting from version 5.1 to version 11, along with the existing features. That way, the recreation of the WSDC 2017 Bali application with the Ionic Framework version 6 has been successfully built run on Android devices version 5.1 to version 11.

Keywords: WSDC, Ionic Framework, UI Component, Capacitor, Cordova, Angular

Untuk Mimih dan Pipih, keluarga, dan diri saya sendiri.

KATA PENGANTAR

Puji dan syukur saya panjatkan kepada Allah SWT. atas ridanya sehingga penulis dapat menyelesaikan penyusunan skripsi yang berjudul “Pembuatan Ulang Aplikasi WSDC 2017 Bali Dengan Ionic 5” dengan baik. Skripsi ini disusun untuk memenuhi syarat kelulusan jurusan Teknik Informatik di Fakultas Teknologi Informasi dan Sains, Universitas Katolik Parahyangan. Penulis juga ingin berterimakasih terhadap beberapa pihak yang telah membantu serta memberi dukungan terhadap penulis dalam menyusun skripsi ini, yaitu:

1. Mimih dan pipih, serta keluarga yang selalu memberi motivasi, dukungan, serta doa kepada penulis selama ini.
2. Bapak Pascal Alfadian, Nugroho, M.Comp. sebagai dosen pembimbing yang telah memberikan dukungan serta arahan terhadap penulis sehingga dapat menyelesaikan skripsi ini dengan baik.
3. Bapak Raymond Chandra Putra, M.T. dan Bapak Chandra Wijaya, MT. sebagai dosen penguji yang telah memberikan kritik dan saran kepada penulis.
4. Segenap dosen Teknik Informatika UNPAR yang telah memberikan ilmu dan pelajaran bagi penulis selama berada di UNPAR.
5. Orang terkasih, Astry Destiana Rhosyta, yang selalu mendukung dan menemani penulis selama penggerjaan skripsi ini.
6. Teman-teman Teknik Informatika UNPAR yang telah berbagi ilmu dan memberikan dukungan kepada penulis selama ini.
7. Pihak-pihak lain yang telah membantu dalam pengujian aplikasi WSDC 2017 Bali.

Penulis berharap semoga skripsi ini dapat bermanfaat bagi pembaca dan pihak-pihak yang membutuhkan bagi pengembangan lebih lanjut terkait skripsi ini. . . .

Bandung, November 2022

Penulis

DAFTAR ISI

KATA PENGANTAR	xv
DAFTAR ISI	xvii
DAFTAR GAMBAR	xix
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Tujuan	2
1.4 Batasan Masalah	2
1.5 Metodologi	3
1.6 Sistematika Pembahasan	3
2 LANDASAN TEORI	5
2.1 WSDC 2017 Bali	5
2.2 Angular	6
2.3 Ionic Framework	12
2.3.1 Native API	12
2.3.2 UI Component	18
2.3.3 Migrasi Ionic 3 ke Ionic 6	24
3 ANALISIS	31
3.1 Analisis Sistem Kini dan Sistem Usulan	31
3.2 Tantangan Pengembangan Sistem Usulan	63
4 PERANCANGAN	65
4.1 Perancangan Kelas	65
4.2 Perancangan Struktur HTML	72
5 IMPLEMENTASI DAN PENGUJIAN	75
5.1 Implementasi	75
5.1.1 Lingkungan Implementasi	75
5.1.2 Hasil Implementasi	75
5.2 Pengujian	82
5.2.1 Pengujian Fungsional	82
5.2.2 Pengujian Eksperimental	83
6 KESIMPULAN DAN SARAN	85
6.1 Kesimpulan	85
6.2 Saran	85
DAFTAR REFERENSI	87

A KODE PROGRAM	89
A.1 Komponen Announcements	89
A.2 Komponen Draw	90
A.3 Komponen Home	92
A.4 Komponen Info	94
A.5 Komponen Result	95
A.6 Komponen Schedule	96
A.7 Komponen Venues	98
A.8 Komponen Venues Map	99

DAFTAR GAMBAR

2.1	Aplikasi WSDC 2017 Bali Saat Ini pada Perangkat Android	5
2.2	Aplikasi WSDC 2017 Bali Saat Ini pada Perangkat Android	6
2.3	Desain <i>server-side</i> tradisional [1]	7
2.4	Desain SPA [1]	8
2.5	Proses Pengemasan Aplikasi Cordova [2]	17
3.1	<i>Use Case Diagram</i> Aplikasi WSDC 2017 Bali	31
3.2	Komponen Announcements pada Aplikasi WSDC 2017 Bali	36
3.3	Komponen Draw pada Aplikasi WSDC 2017 Bali	39
3.4	Komponen Home pada Aplikasi WSDC 2017 Bali	41
3.5	Komponen Info pada Aplikasi WSDC 2017 Bali	45
3.6	Komponen <i>Result</i> pada Aplikasi WSDC 2017 Bali	47
3.7	Komponen <i>Schedule</i> pada Aplikasi WSDC 2017 Bali	49
3.8	Komponen <i>Venues</i> pada Aplikasi WSDC 2017 Bali	53
3.9	Komponen <i>Venues Map</i> pada Aplikasi WSDC 2017 Bali	56
3.10	Penggunaan Community Google Maps dan Geolocation pada Aplikasi WSDC 2017 Bali	60
3.11	Penggunaan Community Google Maps pada Aplikasi WSDC 2017 Bali	61
3.12	Arsitektur Komunikasi Server dengan Aplikasi WSDC 2017 Bali	62
4.1	Diagram Kelas Keseluruhan	65
5.1	Tangkapan Layar Halaman Splash Screen Aplikasi WSDC 2017 Bali	77
5.2	Tangkapan Layar Sidemenu Aplikasi WSDC 2017 Bali	77
5.3	Tangkapan Layar Halaman Home Aplikasi WSDC 2017 Bali	78
5.4	Tangkapan Layar Halaman <i>Announcements</i> Aplikasi WSDC 2017 Bali	78
5.5	Tangkapan Layar Halaman <i>Draw</i> Aplikasi WSDC 2017 Bali	79
5.6	Tangkapan Layar Halaman Info Aplikasi WSDC 2017 Bali	79
5.7	Tangkapan Layar Halaman <i>Result</i> Aplikasi WSDC 2017 Bali	80
5.8	Tangkapan Layar Halaman <i>Schedule</i> Aplikasi WSDC 2017 Bali	80
5.9	Tangkapan Layar Halaman <i>Venues</i> Aplikasi WSDC 2017 Bali	81
5.10	Tangkapan Layar Halaman <i>Venues Map</i> Aplikasi WSDC 2017 Bali	81

BAB 1

PENDAHULUAN

1.1 Latar Belakang

World Schools Debating Championships (WSDC) merupakan sebuah turnamen debat Bahasa Inggris tahunan untuk tim-tim tingkat sekolah menengah yang mewakili berbagai negara [3]. Pada awalnya, kompetisi universitas dunia diselenggarakan di Sydney pada bulan Juli 1988. Anggota Federasi Debat Australia menyadari bahwa tidak ada acara serupa untuk siswa sekolah menengah. Namun kejuaraan universitas dunia ini menunjukkan potensi yang sangat besar untuk kompetisi debat internasional yang melibatkan siswa dari seluruh dunia. Pada tahun 1991, kejuaraan diadakan di Edinburgh. Dan sejak saat itu nama World Schools Debating Championships digunakan dan berlangsung hingga saat ini.

WSDC yang diselenggarakan di Bali, Indonesia pada tahun 2017 memiliki sebuah aplikasi bernama WSDC 2017 Bali yang dikembangkan oleh PT DNArtworks Komunikasi Visual menggunakan *framework* Ionic 3 untuk menunjang acara tersebut. Terdapat beberapa fungsi penting di dalam aplikasi ini, diantaranya adalah jadwal untuk kegiatan peserta, berita tentang acara WSDC yang sedang berlangsung, pemberitahuan mengenai kegiatan acara kepada peserta, informasi lokasi dan peta lokasi kegiatan acara yang sedang berlangsung, dan notifikasi untuk peserta.

Ionic Framework merupakan sebuah *framework open source* lintas platform yang digunakan untuk mengembangkan aplikasi *hybrid* yang bekerja pada berbagai macam platform seluler seperti Android, iOS, dan Windows [4]. Aplikasi *hybrid* merupakan sebuah WebApp yang berjalan di dalam sebuah WebView namun dapat menggunakan fitur-fitur yang disediakan oleh perangkat *mobile* [5]. Dalam kata lain, aplikasi *hybrid* merupakan sebuah aplikasi *native* yang menampilkan *web app* di dalam *web view*. Perbedaan *web app* dengan aplikasi *native* adalah dalam hal penggunaan perangkat keras pada perangkat mobile, dimana aplikasi *native* dapat dengan bebas mendapatkan akses penuh penggunaan perangkat keras sedangkan *web app* tidak.

Salah satu aplikasi *hybrid* yaitu Apache Cordova yang dikembangkan oleh Adobe digunakan untuk memecahkan masalah dimana pada saat mengembangkan suatu aplikasi seluler, setiap vendor perangkat lunak memiliki alat-alat yang unik yang hanya dimiliki oleh vendor tersebut, yaitu *Software Development Kit* (SDK) [5]. Karena SDK yang digunakan berbeda-beda tergantung kepada jenis sistem operasi perangkat seluler, maka tidak dapat membuat aplikasi untuk platform yang berbeda, namun dengan baris kode yang sama. Apache Cordova sebagai aplikasi *hybrid* dapat membuat aplikasi berbasis web seperti HTML, *Cascading Style Sheets* (CSS), dan Javascript, dikemas sebagai aplikasi *native* yang dapat mengakses fitur-fitur perangkat keras dari suatu perangkat.

Ionic Framework mendukung komunikasi dengan menggunakan Native API, yaitu pengembangan aplikasi langsung terintegrasi ke dalam platform [6]. Cordova merupakan Native API yang digunakan untuk menambahkan fungsionalitas ke dalam aplikasi Ionic apapun. Selain Cordova, terdapat Native API lain yang didukung oleh Ionic Framework, yaitu Capacitor. Capacitor merupakan penerus dari Cordova yang menyediakan akses ke perangkat *native* dan fitur platform, serta untuk menyediakan satu set API untuk mengembangkan aplikasi seluler secara *hybrid*, *Progressive Web Apps* berbasis web, dan aplikasi komputer berbasis Electron [7]. Ionic juga memiliki berbagai macam *front-end library* dan *User Interface*(UI) *Components* berupa *tag* khusus yang digunakan untuk menambah fungsionalita aplikasi.

Pada Ionic 5 ke atas, terdapat beberapa *framework Javascript* yang dapat diimplementasikan menggunakan *framework Ionic*, yaitu:

- Angular

Angular pada awalnya diciptakan oleh karyawan Google, Misko Hevert dan Adam Abrons pada tahun 2008, yang masih bernama AngularJS dan dikembangkan dengan JavaScript [8]. Pada saat AngularJS pertama kali diciptakan, sebagian besar situs web menggunakan aplikasi multi-halaman, yaitu ketika pengguna mengklik tautan, maka browser harus mengambil dokumen HTML yang diminta dari server. Angular tidak mengimplementasi hal tersebut, melainkan menggunakan *Single-page Application* (SPA), yaitu ketika halaman awal dimuat, semua yang dibutuhkan untuk membuat dan menampilkan sebuah halaman diunduh, kemudian ditampilkan kedalam layar. Dengan begitu, *browser* tidak perlu mengunduh ulang yang dibutuhkan saat menampilkan halaman [1].

- React

React adalah *library JavaScript open source* untuk membangun antarmuka pengguna, dikelola oleh Facebook, dapat digunakan dalam berbagai skenario termasuk aplikasi iOS dan Android [8].

- Vue

Vue merupakan *framework* progresif untuk membangun antarmuka pengguna untuk web, yang dapat digunakan baik untuk projek kecil dan untuk *Single-Page Applications* (SPAs) [8].

Aplikasi WSDC 2017 Bali yang dibangun pada tahun 2017 oleh PT DNArtworks Komunikasi Visual untuk perangkat iOS untuk sistem operasi iOS telah diturunkan dari App Store dikarenakan tidak mengalami pembaruan dalam jangka waktu tertentu. Maka dari itu diperlukan pembaruan pada aplikasi WSDC 2017 Bali. Pembaruan tersebut dilakukan dengan memperbarui versi Ionic Framework yang sebelumnya versi 3, menjadi versi 6. Pada awalnya pembaruan hanya sampai Ionic Framework versi 5, namun karena Ionic Framework versi 6 diluncurkan pada skripsi ini dibuat, maka dari itu pembaruan dilanjutkan ke Ionic Framework versi 6. Pembaruan Ionic Framework diperlukan karena Ionic versi 3 sudah tidak mendapat dukungan lagi dari tim pengembang Ionic Framework. Untuk melakukan pembaruan tersebut, maka pada skripsi ini dibuat sebuah aplikasi WSDC 2017 Bali baru, yang merupakan sebuah pembaruan dari aplikasi WSDC 2017 Bali. Pembaruan dilakukan dengan cara membuat aplikasi WSDC 2017 Bali baru menggunakan *framework* Ionic versi 6 dan Capacitor dengan fitur-fitur dan halaman yang sama seperti aplikasi sebelumnya. Dilakukan juga penyesuaian pembaruan yang terjadi pada saat melakukan pembaruan dari Ionic 3 ke Ionic 6 serta pembaruan pada saat mengganti Cordova dengan Capacitor.

1.2 Rumusan Masalah

Rumusan masalah yang dibahas pada skripsi ini yaitu:

1. Bagaimana melakukan migrasi aplikasi Android WSDC 2017 Bali ke *framework* Ionic versi 6?
2. Bagaimana menjalankan aplikasi WSDC 2017 Bali pada perangkat Android setelah dilakukan migrasi?

1.3 Tujuan

Tujuan yang ingin dicapai dari penulisan skripsi ini yaitu:

1. Melakukan migrasi aplikasi Android WSDC 2017 Bali ke *framework* Ionic versi 6.
2. Menjalankan aplikasi WSDC 2017 Bali pada perangkat Android setelah dilakukan migrasi.

1.4 Batasan Masalah

Dalam skripsi ini dibuat batasan-batasan masalah dalam pembuatan perangkat lunak. Batasan-batasan masalah yang ditetapkan adalah sebagai berikut:

1. Aplikasi ini tidak akan memiliki fitur notifikasi, dikarenakan sudah tidak terdapat *service* dari Ionic dan tidak dikembangkan lagi dari sisi severnya.
2. Walaupun Ionic Framework mendukung pengembangan aplikasi untuk sistem operasi iOS, aplikasi hanya akan diuji pada *platform mobile* berbasis android.

1.5 Metodologi

Langkah-langkah yang dilakukan dalam skripsi ini adalah sebagai berikut:

1. Melakukan studi mengenai *framework* Ionic versi 3 dan versi 6.
2. Menganalisis aplikasi WSDC 2017 Bali.
3. Mempelajari bagaimana cara melakukan migrasi Ionic versi 3 ke versi 6.
4. Membangun aplikasi WSDC dengan *framework* Ionic versi 6.
5. Melakukan pengujian dan eksperimen.
6. Menulis dokumen skripsi.

1.6 Sistematika Pembahasan

Sistematika penulisan setiap bab pada skripsi ini adalah sebagai berikut:

1. Bab Pendahuluan
Bab 1 berisi latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi, dan sistematika pembahasan yang digunakan untuk menyusun skripsi ini.
2. Bab Dasar Teori
Bab 2 berisi teori-teori yang digunakan dalam pembuatan skripsi ini. Teori-teori tersebut yaitu WSDC, Angular, Ionic Framework, Capacitor, Cordova, UI Components, dan Migrasi Ionic.
3. Bab Analisis
Bab 3 berisi analisis yang dilakukan pada skripsi ini, meliputi analisis sistem kini, analisis kebutuhan aplikasi WSDC 2017 Bali yang akan dibangun, serta permasalahan pembangunan sistem usulan.
4. Bab Perancangan
Bab 4 berisi perancangan aplikasi meliputi perancangan kelas beserta dengan diagram kelas, deskripsi kelas dan fungsinya, serta perancangan struktur HTML.
5. Bab Implementasi dan Pengujian
Bab 5 berisi implementasi dan pengujian aplikasi meliputi lungkungan implementasi, hasil implementasi, pengujian fungsional, dan pengujian eksperimental.
6. Bab Kesimpulan dan Saran
Bab 6 berisi kesimpulan dari hasil pembangunan aplikasi ini dan saran untuk pengembangan selanjutnya.

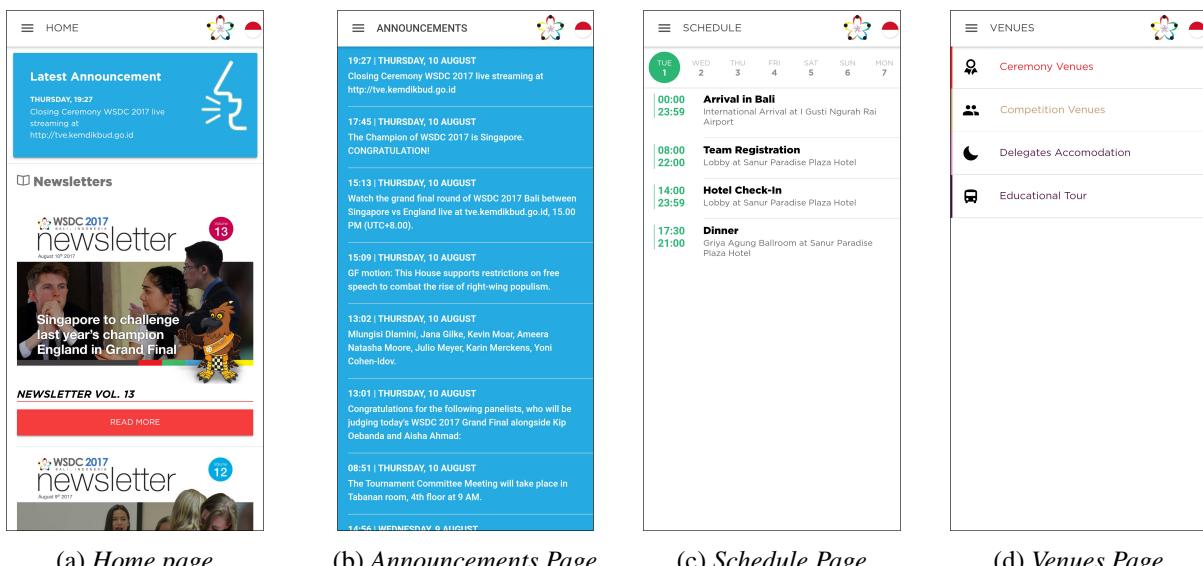
BAB 2

LANDASAN TEORI

Pada bab ini menjelaskan dasar-dasar teori mengenai Ionic, berikut dengan cara untuk melakukan migrasi dari Ionic 3 ke Ionic 6. Akan dibahas pula aplikasi WSDC 2017 Bali saat ini, Angular, Native API berupa Capacitor dan Cordova, dan UI Components.

2.1 WSDC 2017 Bali

Aplikasi WSDC 2017 Bali digunakan untuk menunjang keberlangsungan acara WSDC 2017 yang diselenggarakan di Bali, Indonesia. Aplikasi WSDC 2017 Bali dapat diunduh untuk sistem operasi *android* melalui URL <https://play.google.com/store/apps/details?id=org.wsdc2017indonesia.app>. Aplikasi ini dibangun dan dikembangkan oleh PT DNArtworks Komunikasi Visual yang rilis di Play Store pada tanggal 30 Juli 2017, dengan versi terakhir adalah versi 1.1.2 yang rilis pada 1 Agustus 2017. Selain rilis pada perangkat *android*, aplikasi ini juga rilis untuk perangkat bergerak berbasis sistem operasi iOS. Namun saat skripsi ini dibuat, aplikasi yang dibuat pada tahun 2017 tersebut sudah diturunkan dari App Store pada perangkat berbasis sistem opearsi iOS, dikarenakan tidak mengalami pembaruan dalam jangka waktu tertentu. Untuk membuka dan memakai aplikasi WSDC 2017 Bali saat ini, pengguna tidak diperlukan *login* agar dapat mengakses seluruh fitur yang tersedia. Untuk kepentingan skripsi ini, peneliti memiliki akses ke dalam kode program aplikasi WSDC 2017 Bali.

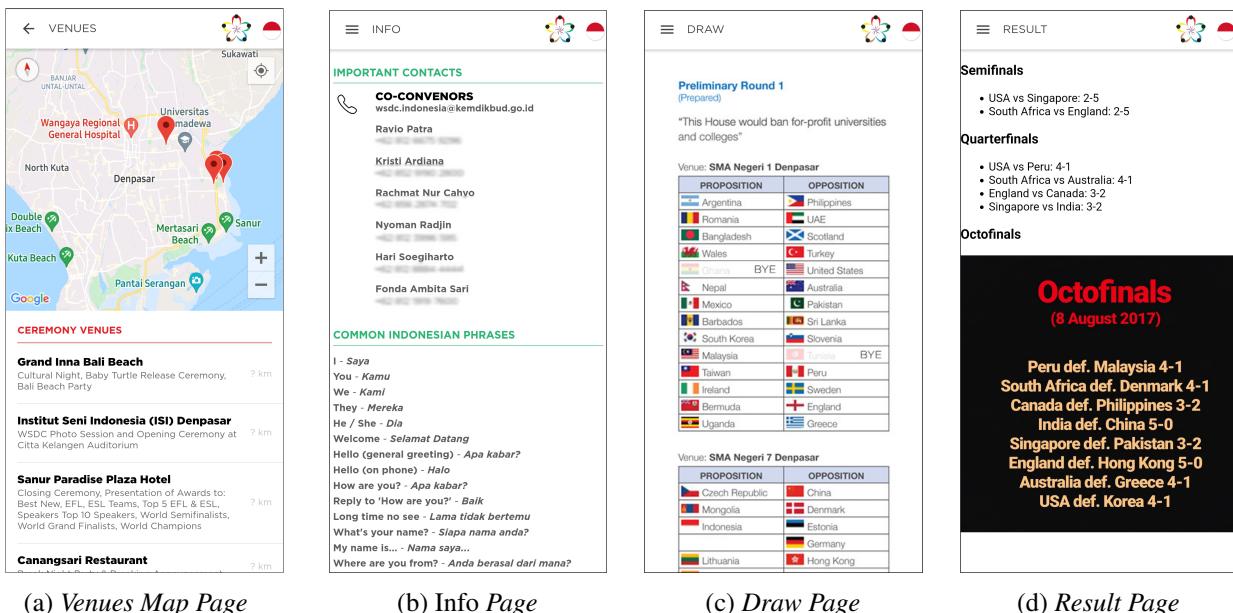


Gambar 2.1: Aplikasi WSDC 2017 Bali Saat Ini pada Perangkat Android

Fitur-fitur yang terdapat di aplikasi WSDC 2017 Bali saat ini yaitu :

1. *Home Page*: Pengguna dapat melihat pengumuman terbaru, dan *headline* dari berita-berita terkait acara WSDC 2017 Bali dengan tombol yang dapat diklik untuk melihat berita tersebut secara lebih detail (Gambar 2.1a).

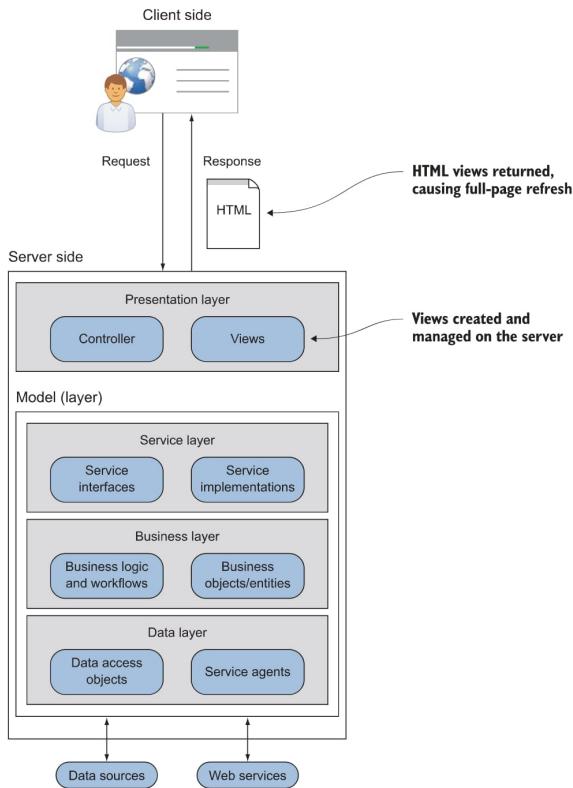
2. *Announcements*: Pengguna dapat melihat pemberitahuan tentang berjalannya acara WSDC 2017 Bali (Gambar 2.1b).
3. *Schedule*: Pengguna dapat melihat jadwal acara WSDC 2017 Bali yang sudah diadakan (Gambar 2.1c).
4. *Venues*: Pengguna dapat melihat berbagai macam lokasi acara WSDC 2017 Bali, mulai dari lokasi upacara, lokasi kompetisi, dan lokasi wisata edukasi (Gambar 2.1d). Masing-masing dari lokasi tersebut ditunjukkan dengan tanda merah pada peta dan dapat melihat jarak dari lokasi perangkat pengguna ke lokasi *venues* (Gambar 2.2a).
5. *Info*: Pengguna dapat melihat informasi terkait dengan tim pengembang dari aplikasi WSDC 2017 Bali, kontak-kontak penting yang dapat dihubungi, dan kosa kata penting dalam Bahasa Indonesia (Gambar 2.2b).
6. *Draw*: Pengguna dapat melihat melihat pembagian *venue* dan kubu proposisi atau oposisi dari hasil pengundian untuk para negara peserta WSDC 2017 Bali (Gambar 2.2c).
7. *Result*: Pengguna dapat melihat informasi terkait hasil dari pertandingan pada semi final, perempat final, dan perdelapan final WSDC 2017 Bali (Gambar 2.2d).



Gambar 2.2: Aplikasi WSDC 2017 Bali Saat Ini pada Perangkat Android

2.2 Angular

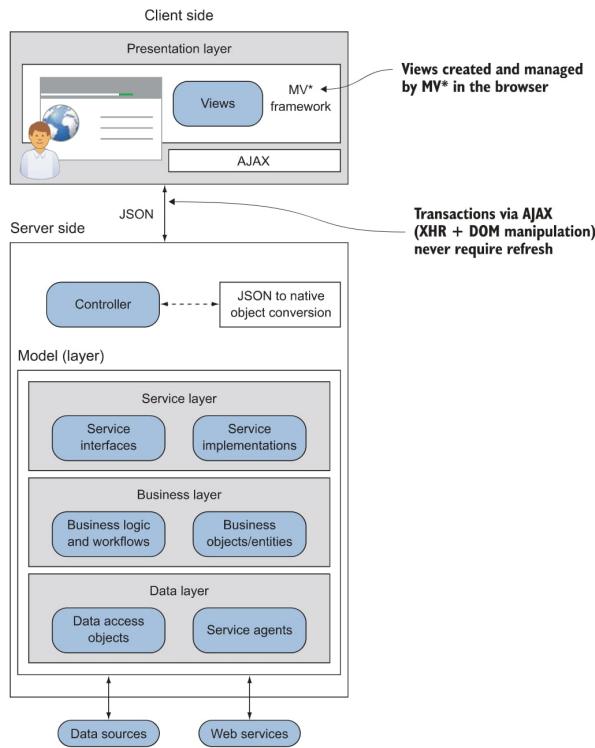
Angular merupakan *framework* Javascript terbuka yang dikembangkan oleh Google, dan merupakan penerus dari versi sebelumnya yaitu AngularJS [9]. Angular versi pertama dirilis pada tahun 2016 dengan nama Angular 2. Aplikasi Angular dapat dibangun dengan menggunakan JavaScript, atau TypeScript.



Gambar 2.3: Desain *server-side* tradisional [1]

Angular dapat digunakan untuk membuat aplikasi Single-page-application (SPA), yaitu ketika halaman awal dimuat, semua yang dibutuhkan untuk membuat dan menampilkan sebuah halaman diunduh, kemudian ditampilkan kedalam layar [1]. Website dengan desain SPA berbeda dengan desain tradisional. Seperti pada Gambar 2.3, pada desain tradisional setiap permintaan untuk tampilan baru, yaitu halaman HTML, memerlukan *request* ke server dan mengembalikannya kembali ke *client*. Ketika data baru diperlukan oleh *client*, permintaan dikirim ke sisi server. Di sisi server, permintaan diambil oleh *controller* di dalam *presentation layer*.

Controller kemudian berinteraksi dengan *model layer* melalui *service layer* untuk menentukan data yang diperlukan. Permintaan terhadap data tersebut diteruskan ke *data layer* yang kemudian mengambil data dari *web service*. Setelah data diambil, setiap perubahan yang diperlukan pada data kemudian dibuat oleh *business logic* di *business layer*. Lalu setelah perubahan yang diperlukan pada data telah selesai di *business logic*, data dikembalikan ke *presentation layer*. Di dalam *presentation layer* menentukan bagaimana data yang baru diperoleh direpresentasikan ke dalam tampilan yang dipilih. Setelah data dan tampilan digabungkan, tampilan dikembalikan ke browser. Browser kemudian menerima halaman HTML setelah melakukan *refresh* pada tampilan antarmuka *browser*. Pada akhirnya pengguna melihat tampilan baru yang berisi data yang diminta setelah *browser* melakukan *refresh*.



Gambar 2.4: Desain SPA [1]

Pada desain SPA, pengelolaan tampilan antarmuka dan server terpisah seperti pada Gambar 2.4. Hal tersebut membuat desain SPA hampir sama dengan desain tradisional, namun terdapat perbedaan berupa tidak ada lagi *refresh* pada *browser* untuk mendapatkan tampilan halaman dengan data yang baru, dan *presentation layer* berada di *client* yang membuat tugas menggabungkan HTML dan data dipindahkan dari server ke browser. Pada desain SPA, setelah halaman awal dimuat, semua data yang diperlukan untuk membuat dan menampilkan tampilan diunduh dan siap digunakan. Jika diperlukan tampilan baru, tampilan tersebut dibuat secara lokal di browser dan ditampilkan secara dinamis melalui JavaScript sehingga tidak diperlukan *refresh* pada browser.

Angular dapat dibangun dengan menggunakan JavaScript atau TypeScript. Namun, penggunaan TypeScript saat ini menjadi lebih produktif dibandingkan dengan JavaScript [9]. TypeScript mengikuti perkembangan terakhir dari ECMAScript, yaitu bahasa skrip yang distandardisasi oleh Ecma International dalam spesifikasi ECMA-262 dan ISO/IEC 16262 [10]. TypeScript juga menambahkan *types*, *interface*, *decorators*, *class member variables*, *generic*, *enum*, dan *keyword* seperti *public*, *protected*, dan *private* serta dapat digunakan untuk mendeklarasikan jenis tipe khusus yang dapat dikustomisasi. Maka dari itu, Framework Angular sendiri ditulis menggunakan TypeScript.

Angular terdiri dari komponen-komponen yang merupakan sebuah penyusun utama untuk aplikasi Angular. Setiap komponen yang ada pada aplikasi, secara *default* memiliki *critical files* yang terdiri dari *file* HTML untuk mendeklarasi halaman yang akan dimuat, *file* TypeScript, serta *file* css [11]. Di dalam komponen terdapat *module.ts* yang merupakan *NgModule* dari sebuah komponen, *spec.ts* yang digunakan untuk menguji komponen, dan *routing.module.ts* yang berisi definisi untuk bernavigasi antar bagian dalam aplikasi Angular. Penjelasan untuk masing-masing *file* adalah sebagai berikut:

- *File routing.module.ts*

File ini digunakan untuk melakukan navigasi antar komponen. Untuk melakukannya, harus melakukan *import* *RouterModule* dan *Routes* sehingga dapat memiliki fungsionalitas *routing* (Kode 2.1). Pada kode tersebut, dilakukan *import* *RouterModule* dan *Routes* dari *@angular/router*. Selanjutnya lakukan *import* untuk komponen lain yang akan dituju untuk melakukan navigasi ke komponen tersebut. Sebagai contoh, pada kode 2.1 melakukan *import* untuk komponen “Heroes”.

```

1 import { RouterModule, Routes } from '@angular/router';
2 import { HeroesComponent } from './heroes/heroes.component';

```

Kode 2.1: Contoh *import* pada routing.module.ts

Setelah itu, lakukan konfigurasi *routes*. Routes kemudian memberi tahu Router tampilan mana yang akan ditampilkan saat pengguna melakukan navigasi. Seperti pada kode 2.2, dengan menggunakan key path dan component. Path digunakan untuk melakukan navigasi di alamat url sesuai dengan path tersebut. Sedangkan component digunakan untuk menampilkan komponen yang dituju.

```

1 const routes: Routes = [
2   { path: 'heroes', component: HeroesComponent }
3 ];

```

Kode 2.2: Contoh Konfigurasi *Routes* pada routing.module.ts

- *File CSS*

File ini digunakan sebagai *template* CSS untuk sebuah komponen. Aplikasi angular ditata dengan CSS standar, yang dapat menerapkan semua CSS stylesheets, *selectors*, *rules*, dan *media queries* langsung ke aplikasi Angular.

- *File HTML*

File HTML pada Angular sama seperti HTML biasa, dan bisa ditambahkan dengan sintaks Angular. HTML pada komponen digunakan untuk mendeklarasi halaman yang akan dimuat.

- *File specs.ts*

File ini digunakan untuk melakukan pengujian terhadap aplikasi Angular.

- *File TypeScript*

Di dalam TypeScript, terdapat sebuah kelas komponen. Kelas tersebut memiliki anotasi dengan sebuah *decorator* (Kode 2.3) yang ditempatkan sesuai dengan komponen UI nya berada. Komponen berisi *instance* dari service class yang diimplementasi dari *business logic* tanpa UI. Sebuah service class merupakan implementasi dari *business logic*. Angular akan menempatkan *services* ke dalam komponen atau ke dalam *services* lainnya dengan menggunakan *dependency injection* (DI).

```

1 @Component({
2   ...
3 }
4 export class AppComponent {
5   ...
6 })

```

Kode 2.3: Anotasi Komponen dengan *Decorator*

Pada Angular terdapat decorator `@Component` yang digunakan untuk memanggil HTML *template* dan CSS untuk komponen. Pada setiap komponen terdapat HTML *template* yang berada di dalam komponen tersebut, atau di dalam file yang berada di luar file komponen yang direferensikan dengan menggunakan properti `templateUrl` di dalam komponen pada file TypeScript. File HTML *template* yang terpisah dengan komponen, memberikan efek kode yang lebih bersih di dalam komponen nya. Selain HTML *template*, file lain seperti file CSS dapat diletakkan terpisah dari file komponen dengan menggunakan `styleUrls` untuk mereferensikan file CSS ke dalam komponen (Kode 2.4). Lalu terdapat juga properti selector untuk mendefinisikan nama dari tag yang bisa digunakan atau dipanggil oleh komponen lain.

```

1 @Component({
2   selector: 'app-search',
3   templateUrl: './search.component.html',
4   styleUrls:[ './search.component.css' ]
5 })
6 export class SearchComponent {
7   ...

```

8 }

Kode 2.4: Contoh Mereferensikan HTML dan CSS di Dalam Komponen

- *File module.ts*

Komponen-komponen yang ada akan dikelompokkan menjadi Angular modules, yaitu sebuah kelas yang diberi `@NgModule()`. Angular module biasanya merupakan sebuah kelas kecil yang tidak memiliki isi, kecuali menulis kode bootstrap secara manual ke dalam aplikasi. Contohnya, ketika sebuah aplikasi merupakan turunan dari AngularJS yang lama, *decorator* `@NgModule()` menampilkan semua komponen dan *artifacts* lain termasuk *services*, *directive*, dan yang lainnya, maka semua itu harus disertakan ke dalam module (Kode 2.5).

```

1 @NgModule({
2   declarations: [
3     AppComponent
4   ],
5   imports: [
6     BrowserModule
7   ],
8   bootstrap: [AppComponent]
9 })
10 export class AppModule{
11   ...
12 }
```

Kode 2.5: Module dengan Komponen

Di dalam aplikasi Angular, terdapat beberapa komponen, seperti Parent Component, dan Child Component. Parent Component dapat mengirimkan data ke properti Child Component-nya. Namun, Child Component tidak tahu siapa yang mengirimkannya. Sedangkan Child Component juga dapat mengirimkan data ke Parent Component, meskipun dia tidak tahu siapa Parent Component-nya. Arsitektur seperti ini dapat membuat komponen menjadi mandiri dan dapat digunakan kembali.

Angular menyediakan beberapa *libraries*, yaitu:

1. Angular Router

Angular Router digunakan untuk menangani navigasi dari satu tampilan ke tampilan lain, dan kemudian menampilkannya di layar [12]. Langkah-langkah untuk membuat *route* pada Angular adalah sebagai berikut:

- (a) Import RouterModule dan Routes ke routing module.

Pada tahap ini, Angular CLI membuatnya secara otomatis saat pembuatan proyek Angular. Angular CLI membuat array Routes, dan melakukan import serta export array di `@NgModule()`. Seperti pada contoh kode 2.6 pada baris ke-2 dilakukan import untuk Routes dan RouterModule dari Angular, serta pada baris ke-4 Angular CLI membuat array Routes.

```

1 import { NgModule } from '@angular/core';
2 import { Routes, RouterModule } from '@angular/router';
3
4 const routes: Routes = [];
5
6 @NgModule({
7   imports: [RouterModule.forRoot(routes)],
8   exports: [RouterModule]
9 })
10 export class AppRoutingModule { }
```

Kode 2.6: Contoh Routing pada Angular

- (b) Menambahkan *routes* di array Routes.

Tahap selanjutnya adalah memasukan *route* pada array routes yang telah dibuat pada tahap sebelumnya. Seperti pada kode 2.7, terdapat properti path dan component pada setiap elemen di array.

Properti path digunakan untuk path URL untuk route yang dituju. Properti component digunakan untuk menunjukkan komponen apa yang digunakan. Sebagai contoh pada kode 2.7 baris ke-2, properti path berisi komponen 'first-component' yang komponennya berada di FirstComponent pada properti component. Ketika pengguna memnulis path 'first-component' (seperti pada properti path) di dalam URL, maka komponen FirstComponent pada properti component yang akan dipanggil yang kemudian menampilkan komponen first-component.

```

1 const routes: Routes = [
2   { path: 'first-component', component: FirstComponent },
3   { path: 'second-component', component: SecondComponent },
4 ];

```

Kode 2.7: Contoh *Routing* pada Angular

(c) Menambahkan Routes ke Aplikasi.

Selanjutnya, untuk mengakses halaman sesuai dengan *routes*-nya, yaitu dengan memanggil properti path pada tahap sebelumnya di dalam routerLink seperti pada kode 2.8 baris ke-4. RouterLink digunakan untuk membuka komponen ketika pengguna melakukan klik. Setelah pengguna melakukan klik, maka routerLink akan memanggil routes dengan path first-component dan mengembalikan komponen FirstComponent.

```

1 <h1>Angular Router App</h1>
2 <nav>
3   <ul>
4     <li><a routerLink="/first-component" routerLinkActive="active"
ariaCurrentWhenActive="page">First Component</a></li>
5     <li><a routerLink="/second-component" routerLinkActive="active"
ariaCurrentWhenActive="page">Second Component</a></li>
6   </ul>
7 </nav>
8 <router-outlet></router-outlet>

```

Kode 2.8: Contoh *Routing* pada Angular

2. Angular HTTPClient.

Angular HTTPClient digunakan untuk melakukan komunikasi dengan server dengan menggunakan protokol HTTP, baik itu melakukan *download* atau *upload* data. Untuk dapat menggunakan HTTPClient, pertama harus melakukan import pada AppModule di file app.module.ts. Seperti pada kode 2.9, pada baris ke-3 dilakukan import HttpClientModule. Kemudian lakukan import pada @NgModule seperti pada baris ke-8.

```

1 import { NgModule } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-browser';
3 import { HttpClientModule } from '@angular/common/http';
4
5 @NgModule({
6   imports: [
7     BrowserModule,
8     HttpClientModule,
9   ],
10  declarations: [
11    AppComponent,
12  ],
13  bootstrap: [ AppComponent ]
14})
15 export class AppModule {}

```

Kode 2.9: Contoh *Routing* pada Angular

HTTPClient memiliki sebuah *method* `get()` yang digunakan untuk mengambil data dari server. *Method* ini mengirim HTTP Request, dan mengembalikan sebuah data JSON yang ada pada *response body*. Kode 2.10 adalah contoh untuk mengambil data dari server dengan menggunakan HTTPClient. Pada kode tersebut, terdapat *method* `subscribe()` seperti pada baris ke-4 yang membuat HTTPClient menulis dan mengirim HTTP Request ke server yang kemudian mendapatkan data untuk digunakan di dalam aplikasi.

```

1 const testData: Data = {name: 'Test Data'};
2
3 httpClient.get<Data>(testUrl)
4 .subscribe(data =>
5   expect(data).toEqual(testData)
6 );
7
8 const req = httpTestingController.expectOne('/data');
9
10 expect(req.request.method).toEqual('GET');
11
12 req.flush(testData);
13
14 httpTestingController.verify();

```

Kode 2.10: Contoh *Routing* pada Angular

2.3 Ionic Framework

Ionic Framework merupakan sebuah *framework open source* lintas platform yang memungkinkan untuk mengembangkan aplikasi hibrida yang bekerja pada berbagai macam platform seluler seperti *android*, *iOS*, dan *Windows* [4]. Ionic memiliki berbagai macam *front-end library* dan komponen *User Interface*(UI) yang digunakan untuk perancangan aplikasi menggunakan teknologi web seperti HTML, CSS, dan Javascript, dengan integrasi untuk berbagai *framework* seperti Angular, React, dan Vue. Saat pertama kali dibuat, Ionic menggunakan AngularJS. Namun, pada saat Angular versi 2 yang menggunakan Typescript dirilis, Ionic versi 2 dan selanjutnya menggunakan Angular. Pada tahun 2019, Ionic mendukung penggunaan *framework* lain selain Angular, yaitu React dan Vue. Di dalam Ionic, Angular digunakan untuk membangun aplikasi dan perutean, sehingga aplikasi dapat sejalan dengan ekosistem Angular lainnya. Ionic menyediakan *toolkit* Angular untuk membangun aplikasi dan terintegrasi dengan Angular CLI resmi yang menyediakan fitur khusus untuk aplikasi Ionic Angular. Pada saat skripsi ini dibuat, Ionic versi terbaru adalah Ionic versi 6, dengan dukungan penggunaan Angular versi 12 sampai dengan yang lebih baru.

2.3.1 Native API

Native API memungkinkan pengembangan aplikasi langsung terintegrasi ke dalam platform. Pengembang dapat membuat aplikasi pada perangkat *mobile* untuk dapat diimplementasikan ke berbagai *platform*, seperti *iOS* dan *Android*, setelah pengembangan selesai di dalam *framework native* tanpa perlu perubahan, dan tidak mempengaruhi peforma dari aplikasi tersebut [6].

Ionic mendukung komunikasi dengan menggunakan Native API yang terintegrasi untuk menambahkan fungsionalitas ke dalam aplikasi Ionic apapun dengan menggunakan Capacitor atau Cordova. Dengan terpasangnya Ionic Native, maka aplikasi akan memiliki antar muka yang diperlukan untuk berinteraksi dengan salah satu *plug-in*, yaitu Capacitor atau Cordova.

2.3.1.1 Capacitor

Capacitor adalah Native API yang bertujuan untuk menyediakan akses ke dalam fitur-fitur perangkat *mobile*, serta untuk menyediakan satu set API untuk mengembangkan aplikasi seluler secara *hybrid*, *Progressive Web Apps* berbasis web, dan aplikasi komputer berbasis Electron [7]. Capacitor merupakan penerus dari Cordova, dengan tujuan untuk memungkinkan aplikasi web modern berjalan di semua platform utama. Capacitor juga mendapat dukungan terhadap banyak *plugin* Cordova.

Capacitor digunakan untuk membuat aplikasi secara hybrid yang memungkinkan pembuatan aplikasi seluler untuk beberapa platform dengan menggunakan baris kode yang sama [13]. Karena aplikasi yang dibangun dengan Ionic Framework merupakan sebuah aplikasi web (WebApp), maka dari itu untuk menjalankan WebApp tersebut secara langsung dari platform mobile dibutuhkan sebuah penghubung antara WebApp dengan perangkat, yaitu Capacitor. Capacitor menjadi sebuah *native bridge* yang menjembatani antara WebView dengan siklus hidup terakhir pada program yaitu Runtime.

Aplikasi WebApp menjalankan URL, kemudian WebView akan ditampilkan sama seperti saat membuka *browser*, tapi dengan Capacitor maka WebApp tersebut memiliki akses ke berbagai fitur *native* perangkat seperti kamera, notifikasi, GPS, dan file sistem. Untuk mengakses fitur-fitur tersebut dibutuhkan plugin. Capacitor memiliki *plugins* untuk mengakses fitur-fitur tersebut. *Plugins* yang dimiliki oleh Capacitor terbagi menjadi dua, yaitu:

1. *Official Plugins*

Official Plugins merupakan sekumpulan *plugin* resmi yang dikelola oleh tim Capacitor untuk menyediakan akses ke *native API* suatu perangkat. Terdapat beberapa *plugin* pada *Official Plugins*, diantaranya adalah sebagai berikut:

- (a) Browser API

Browser API menyediakan kemampuan untuk membuka browser dalam aplikasi. Untuk menginstal Browser API dapat dilakukan melalui *command line* dengan perintah seperti pada kode 2.11.

```
1 npm install @capacitor/browser
2 npx cap sync
```

Kode 2.11: Kode untuk Instalasi Browser API

Browser API memiliki beberapa *method*, yaitu:

- *open()*: *method* ini digunakan untuk membuka halaman *browser* dengan opsi yang sudah ditentukan.
- *close()*: *method* ini digunakan untuk menutup halaman *browser* yang sedang dibuka. *Method* ini hanya dapat digunakan pada web dan iOS.
- *addListener('browserFinished', ...)*: *method* ini merupakan sebuah *listener* untuk *browser finished event* dan dipanggil ketika *browser* ditutup oleh pengguna. *Method* ini hanya dapat digunakan pada Android dan iOS.
- *addListener('browserPageLoaded', ...)*: *method* ini merupakan sebuah *listener* untuk *page loaded event* dan aktif ketika URL diteruskan ke *method finished loading*. *Method* ini hanya dapat digunakan pada Android dan iOS.
- *removeAllListeners()*: *method* ini digunakan untuk menghapus semua *native listeners* untuk *plugin* ini.

- (b) Geolocation API

Geolocation API menyediakan *method* untuk mendapatkan dan melacak posisi perangkat saat ini menggunakan *Global Positioning System* (GPS), dengan latitude dan longitude, juga informasi mengenai ketinggian, arah, dan kecepatan jika tersedia. Untuk menginstal Geolocation API dapat dilakukan melalui *command line* dengan perintah seperti pada kode 2.12.

```
1 npm install @capacitor/geolocation
2 npx cap sync
```

Kode 2.12: Kode untuk Menginstal Geolocation API

Pada perangkat Android, dibutuhkan izin untuk menggunakan Geolocation API 2.13. Izin tersebut ditambahkan ke dalam baris kode pada file `AndroidManifest.xml`. Izin tersebut digunakan untuk meminta data lokasi (*fine* dan *coarse*), dan izin untuk menggunakan GPS. Dengan menggunakan GPS, Geolocation API akan mengembalikan koordinat dari perangkat yang berupa latitude dan longitude. Contoh dari penggunaan Geolocation API untuk mengambil koordinat perangkat tertera pada kode 2.14.

```
1 <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
2 <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
3 <uses-feature android:name="android.hardware.location.gps" />
```

Kode 2.13: *Permissions* Geolocation API pada Android

```
1 import { Geolocation } from '@capacitor/geolocation';
2
3 const printCurrentPosition = async () => {
4   const coordinates = await Geolocation.getCurrentPosition();
5
6   console.log('Current position:', coordinates);
7 }
```

Kode 2.14: *Permissions* Geolocation API pada Android

Untuk mengambil posisi dari perangkat dengan menggunakan *method* `getCurrentPosition()` seperti pada kode 2.14. Selain itu terdapat beberapa *method* lain yang dapat diimplementasikan, diantaranya yaitu:

- `watchPosition()`: digunakan untuk mendaftarkan fungsi handler yang akan dipanggil secara otomatis setiap kali posisi perangkat berubah.
- `clearWatch()`: digunakan untuk membatalkan pendaftaran fungsi handler yang sebelumnya diinstal menggunakan `watchPosition()`.
- `checkPermissions()`: digunakan untuk mengecek izin penggunaan lokasi.
- `requestPermissions()`: digunakan untuk meminta izin penggunaan lokasi.

(c) Splash Screen API

Splash Screen API digunakan sebagai penyedia *method* untuk menampilkan atau menyembunyikan gambar *splash*. Untuk menginstal Splash Screen API dapat dilakukan melalui *command line* dengan perintah seperti pada kode 2.15. Contoh penggunaan Splash Screen dapat dilihat pada kode 2.16

```
1 npm install @capacitor/splash-screen
2 npx cap sync
```

Kode 2.15: Kode untuk Menginstal Splash Screen API

```
1 import { SplashScreen } from '@capacitor/splash-screen';
2
3 // Hide the splash (you should do this on app launch)
4 await SplashScreen.hide();
5
6 // Show the splash for an indefinite amount of time:
7 await SplashScreen.show({
8   autoHide: false
9 });
10
11 // Show the splash for two seconds and then automatically hide it:
12 await SplashScreen.show({
13   showDuration: 2000,
14   autoHide: true
15 });
```

Kode 2.16: Contoh Kode Penggunaan Splash Screen API

Sebagai suatu standar, Splash Screen diatur secara otomatis untuk disembunyikan dalam waktu 500ms. Pada kondisi tertentu Splash Screen tidak sepenuhnya menutupi layar, seperti pada bagian-bagian sudut-sudut layar. Splash Screen dapat mengatur warna latar belakang, dibandingkan dengan menampilkan warna transparan saat Splash Screen tidak sepenuhnya menutupi layar. Splash Screen juga dapat dibuat untuk menampilkan *spinner* diatas Splash Screen.

Selain itu, terdapat beberapa *Official Plugins* lain yang dimiliki Capacitor, yaitu Action Sheet, App, App Launcher, Camera, Clipboard, Device, Dialog, Filesystem, Google Maps, Haptics, Keyboard, Local Notifications, Motion, Network, Push Notifications, Screen Reader, Share, Status Bar, Storage, Text Zoom, dan Toast.

2. Community Plugins

Community Plugins merupakan sekumpulan *plugin* yang diciptakan oleh komunitas untuk menambahkan fungsionalitas ke dalam aplikasi. Perbedaan dengan *Official Plugins* yaitu tim Capacitor tidak secara resmi melakukan pemeliharaan terhadap *Community Plugins*. Salah satu *plugin* yang diciptakan oleh komunitas adalah *plugin* Google Maps yang menggunakan *native* Google Maps SDK. Untuk menginstal *plugin* Google Maps dapat dilakukan melalui *command line* dengan perintah seperti pada kode 2.17.

```
1 npm i --save @capacitor-community/google-maps
2 npx cap sync
```

Kode 2.17: Kode untuk Menginstal *Plugin* Google Maps

Maps SDK merender *native element* (MapView) di belakang aplikasi web (WebApp) membutuhkan *boundaries* yang dirender. Maka dari itu Maps SDK menggunakan *native element* dibandingkan HTMLElement. Sebelum dapat menggunakan *plugin* ini, harus dilakukan impor terlebih dahulu seperti pada kode 2.18. Setelah mengimpor *plugin*, sebuah instance Maps sederhana dapat diinisialisasi seperti pada kode 2.19. Pada contoh kode tersebut, Capacitor Community Google Maps memiliki sebuah *function* createMap yang digunakan untuk membuat peta Google Maps. Di dalam *function* tersebut terdapat sebuah *option* boundingRect yang digunakan sebagai lokasi untuk menyimpan peta Google Maps. Karena google maps ditempatkan di dalam sebuah kontainer pada HTML, maka dari itu pada *option* dibutuhkan lokasi *width*, *height*, *x*, dan *y*. Lokasi tersebut diambil dari kontainer yang dibuat pada HTML yang sebelumnya sudah didapatkan pada baris ke-10. Kemudian nilai tersebut dimasukan ke dalam *option* boundingRect pada baris ke-11 sebagai lokasi dari Google Maps.

```
1 import { CapacitorGoogleMaps } from "@capacitor-community/google-maps";
```

Kode 2.18: Kode untuk Import *Plugin* Google Maps

```
1 const initializeMap = async () => {
2   await CapacitorGoogleMaps.initialize({
3     key: "YOUR_IOS_MAPS_API_KEY",
4     devicePixelRatio: window.devicePixelRatio, // this line is very important
5   });
6
7   const element = document.getElementById("container");
8   const boundingRect = element.getBoundingClientRect();
9   try {
10     const result = await CapacitorGoogleMaps.createMap({
11       boundingRect: {
12         width: Math.round(boundingRect.width),
13         height: Math.round(boundingRect.height),
14         x: Math.round(boundingRect.x),
15         y: Math.round(boundingRect.y),
16       },
17     });
18     element.style.background = "";
19     element.setAttribute("data-maps-id", result.googleMap.mapId);
20   }
```

```

21     alert("Map loaded successfully");
22 } catch (e) {
23     alert("Map failed to load");
24 }
25 };
26
27 (function () {
28     initializeMap();
29 })();

```

Kode 2.19: Contoh Kode Penggunaan *Plugin* Google Maps

Capacitor dapat menambahkan platform *native* ke dalam proyek Ionic, baik untuk sistem operasi Android seperti pada kode 2.20 maupun iOS seperti pada kode 2.21. Setelah itu, Capacitor akan melakukan instalasi *package* platform Capacitor, dan menyalin *template native* platform ke dalam proyek. Kemudian, Capacitor juga dapat melakukan pembuatan aplikasi *native* dengan menyalin aset web ke dalam *native* platform dengan perintah seperti pada kode 2.22 untuk perangkat berbasis iOS dan perintah pada kode 2.23 untuk perangkat berbasis Android. Setelah perintah tersebut dieksekusi, maka Capacitor akan membuka IDE dari proyek *native*, seperti Xcode untuk perangkat berbasis iOS dan Android Studio untuk perangkat berbasis Android. Dengan begitu, maka aplikasi dapat berjalan secara *native* di dalam perangkat terkait.

```
1 ionic capacitor add android
```

Kode 2.20: Kode untuk Menambahkan Platform Android dengan Capacitor

```
1 ionic capacitor add ios
```

Kode 2.21: Kode untuk Menambahkan Platform iOS dengan Capacitor

```
1 ionic capacitor build ios
```

Kode 2.22: Kode untuk Membuat Aplikasi Capacitor Untuk Perangkat iOS

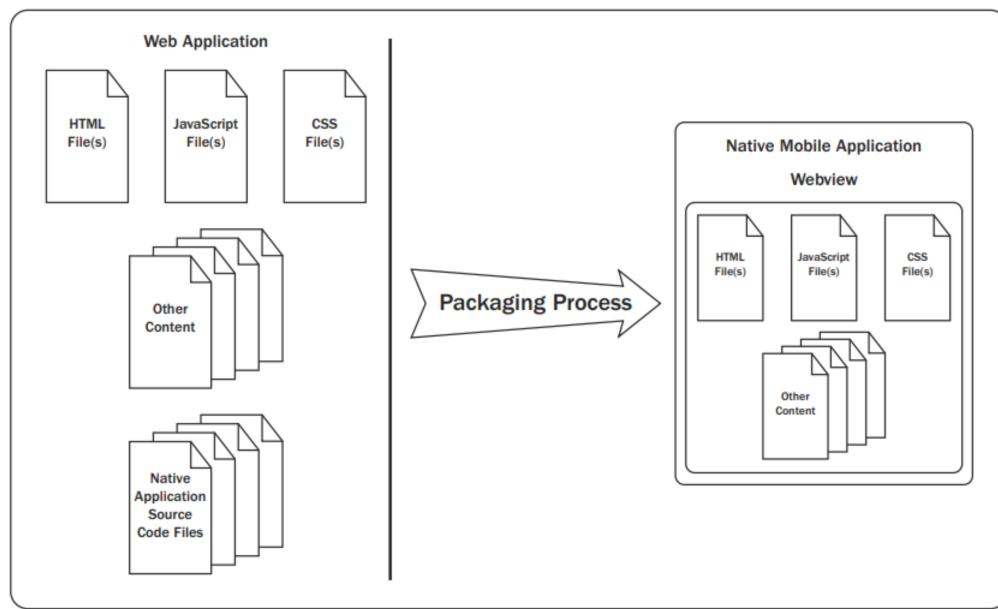
```
1 ionic capacitor build android
```

Kode 2.23: Kode untuk Membuat Aplikasi Capacitor Untuk Perangkat Android

2.3.1.2 Cordova

Cordova merupakan *framework open source* yang dapat membuat pengembang untuk menggunakan teknologi seperti HTML, JavaScript, dan CSS untuk membangun aplikasi untuk perangkat bergerak yang dapat berjalan pada beberapa sistem operasi *mobile* [14]. Cordova menyediakan antarmuka antara WebView dan lapisan *native* pada perangkat [6]. Selain dapat bekerja pada dua platform seluler Android dan iOS, Cordova juga dapat digunakan pada platform seluler seperti Windows Phone, Blackberry, dan FireOS.

Cordova digunakan untuk membangun aplikasi seluler lintas platform dan mengimplementasikannya sebagai kombinasi aplikasi web dan aplikasi *native*, yaitu aplikasi Hybrid [2]. Aplikasi dibuat dengan menggunakan teknologi web seperti HTML dan CSS yang dinamakan WebApp. Cordova mengemas aplikasi web ke dalam *native container*, seperti yang diilustrasikan pada Gambar 2.5.



Gambar 2.5: Proses Pengemasan Aplikasi Cordova [2]

Di dalam aplikasi Cordova, antarmuka pengguna aplikasi terdiri dari satu layar yang hanya berisi satu tampilan web. Saat aplikasi dijalankan, aplikasi memuat halaman awal WebApp ke dalam tampilan web. WebApp yang berjalan di dalam *native container* sama seperti aplikasi web lainnya yang berjalan di dalam browser web pada perangkat *mobile*. WebApp dapat membuka halaman HTML, menjalankan logika JavaScript, dan mengimplementasi CSS, namun WebApp tidak dapat menjalankan fitur-fitur perangkat keras dan perangkat lunak dari perangkat *mobile* seperti akselerometer, kamera, data kontak, dan fitur lainnya. Cordova menyediakan berbagai JavaScript API yang dapat digunakan agar aplikasi web yang berjalan di dalam *native container* dapat mengakses kemampuan perangkat di luar kemampuan *browser*. API ini diimplementasikan ke dalam JavaScript *libraries* yang menjalankan fitur-fitur perangkat ke aplikasi web dengan implementasi yang disesuaikan dengan sistem operasi yang digunakan.

Cordova membuat aplikasi dapat mengakses fitur-fitur perangkat keras dan perangkat lunak suatu perangkat dengan menggunakan *plugin*. Framework Ionic telah terdapat berbagai macam TypeScript *wrapper* untuk *plugins* Cordova. Untuk dapat menggunakan Cordova Plugins, yaitu dengan memasang Cordova Plugins terlebih dahulu yang dapat dipasang dengan menjalankan Kode 2.24, dan memperbaruiya ke versi terakhir pada Kode 2.25 yang dapat dilakukan melalui CLI. Setiap *plugins* memiliki dua komponen, yaitu kode *native* (Cordova), dan kode TypeScript (Ionic Native).

```
1 npm install cordova-plugin-name
2 npx cap sync
```

Kode 2.24: Kode untuk Memasang Cordova Plugins

```
1 npm install cordova-plugin-name@2
2 npx cap update
```

Kode 2.25: Kode untuk Memperbarui Cordova Plugins

Sama seperti Capacitor, Cordova memiliki *plugins* yang menyediakan antarmuka JavaScript ke komponen *native*. Dengan menggunakan *plugin* Cordova, aplikasi dapat menjalankan fitur-fitur perangkat *mobile* seperti kamera, geolokasi, dan file sistem. Salah satu *plugin* yang tersedia yaitu Google Maps. *Plugin* ini menghasilkan MapView secara *native*, dan menempatkannya di bawah browser. MapView yang dihasilkan bukan merupakan HTMLElements, maka tidak terkait dengan HTML. Sebelum dapat menggunakan *plugin* Google Maps, dilakukan instalasi *plugin* untuk pertama kali pada *command line* 2.26. Selanjutnya atur Google Maps API Key di dalam *file config.xml* 2.27.

```
1 cordova plugin add cordova-plugin-googlemaps
```

Kode 2.26: Kode untuk Menginstal *Plugin* Cordova Google Maps

```
1 <widget ...>
2   <preference name="GOOGLE_MAPS_ANDROID_API_KEY" value="(api key)" />
3   <preference name="GOOGLE_MAPS_IOS_API_KEY" value="(api key)" />
4 </widget>
```

Kode 2.27: Kode untuk Mengatur API Key untuk *Plugin* Cordova Google Maps

Cordova juga dapat menambahkan platform *native* ke dalam proyek Ionic dengan mengetikan perintah seperti pada kode 2.28 untuk menambahkan platform Android, dan kode 2.29 untuk menambahkan platform iOS. Untuk menjalankan proyek Ionic dengan Cordova dengan mengetikan perintah seperti pada kode 2.30 untuk perangkat Android, dan kode 2.31 untuk perangkat iOS. Dengan begitu, Cordova akan membuka IDE sesuai dengan perintah yang dijalankan.

```
1 ionic cordova platform add android
```

Kode 2.28: Kode untuk Menambahkan Platform Android dengan Cordova

```
1 ionic cordova platform add ios
```

Kode 2.29: Kode untuk Menambahkan Platform iOS dengan Cordova

```
1 ionic cordova platform run android
```

Kode 2.30: Kode untuk Membuat Aplikasi Cordova Untuk Perangkat Android

```
1 ionic cordova platform run ios
```

Kode 2.31: Kode untuk Membuat Aplikasi Cordova Untuk Perangkat iOS

2.3.2 UI Component

Framework Ionic dapat menggunakan kemampuan Angular dalam memperluas kosakata HTML, yaitu menyertakan *tag* khusus untuk menciptakan seluruh rangkaian komponen [6]. Semua komponen memiliki awalan ion, sehingga dapat dikenali dalam markup. Sama seperti *tag* HTML standar, komponen Ionic juga dapat menerima berbagai macam atribut sebagai pengaturan dari *tag* tersebut, seperti mengatur id atau mendefinisikan kelas CSS tambahan. Terdapat beberapa komponen yang ada pada *framework* Ionic versi 6 [1]. Komponen-komponen tersebut yaitu:

- Action Sheet

Merupakan dialog yang menampilkan serangkaian opsi, yang muncul di atas konten aplikasi dan harus ditutup secara manual oleh pengguna sebelum pengguna dapat melanjutkan interaksi dengan aplikasi. Untuk menutup Action Sheet terdapat beberapa cara, termasuk mengetuk bagian selain Action Sheet atau menekan tombol escape di desktop.

- Alert

Alert merupakan dialog yang menampilkan informasi kepada pengguna, atau mengumpulkan informasi dari pengguna menggunakan input. Alert muncul di atas konten aplikasi, dan harus ditutup secara manual oleh pengguna sebelum pengguna dapat melanjutkan interaksi dengan aplikasi. Secara opsional, terdapat header, sub header, dan pesan yang ada pada Alert.

- Badge

Merupakan elemen *inline block* yang biasanya muncul di dekat elemen lain, berisi angka atau karakter lain, yang digunakan sebagai pemberitahuan bahwa ada item tambahan yang terkait dengan suatu elemen dan menunjukkan berapa banyak item yang ada. Penggunaan Badge dengan menggunakan *tag* <ion-badge> (Kode 2.32).

```
1 <ion-badge>99</ion-badge>
```

Kode 2.32: Potongan Kode Program dari Badge Component

- Button

Merupakan elemen yang dapat diklik, biasanya digunakan dalam formulir atau di mana pun yang membutuhkan fungsionalitas tombol. Button biasanya menampilkan teks, ikon, atau bisa juga keduanya. Button dapat pula menggunakan atribut untuk menampilkannya dengan penampilan tertentu. Penggunaan Button dengan menggunakan tag `<ion-button>` (Kode 2.33).

```
1 <ion-button>Default</ion-button>
```

Kode 2.33: Potongan Kode Program dari Button Component

- Card

Merupakan bagian standar dari tampilan antarmuka, yang tampak seperti kartu yang berfungsi sebagai titik masuk ke dalam informasi yang lebih detail. Card dapat menjadi satu komponen, tetapi sering kali terdiri dari beberapa header, judul, sub judul, dan konten. Penggunaan Card dengan menggunakan tag `<ion-card>` yang dapat berisi *header*, *subtitle*, *title*, dan *content* (Kode 2.34).

```
1 <ion-card>
2   <ion-card-header>
3     <ion-card-subtitle>Card Subtitle</ion-card-subtitle>
4     <ion-card-title>Card Title</ion-card-title>
5   </ion-card-header>
6
7   <ion-card-content>
8     Card Content
9   </ion-card-content>
10 </ion-card>
```

Kode 2.34: Potongan Kode Program dari Card Component

- Content

Komponen content merupakan penyedia area konten yang bisa digunakan untuk mengontrol area yang dapat digulir. Dalam satu tampilan, setidaknya terdapat satu buah content. Content juga dapat dimodifikasi padding, margin, dan lainnya menggunakan *global style* yang berada di CSS Utilities atau mengubahnya secara individual dengan menggunakan CSS. Penggunaan Content dengan menggunakan tag `<ion-content>` (Kode 2.35).

```
1 <ion-content
2   [scrollEvents] = "true"
3   (ionScrollStart) = "logScrollStart()"
4   (ionScroll) = "logScrolling($event)"
5   (ionScrollEnd) = "logScrollEnd()"
6   <h1>Main Content</h1>
7
8   <div slot = "fixed">
9     <h1>Fixed Content</h1>
10    </div>
11 </ion-content>
```

Kode 2.35: Potongan Kode Program dari Content Component

- Date and Time Pickers

Datetime merupakan penampilan antarmuka untuk pengguna memilih tanggal dan waktu. Terdapat kolom yang dapat digulir yang dapat digunakan untuk memilih tahun, bulan, hari, jam, dan menit secara individual. Komponen ini menampilkan nilai di dua tempat, yaitu di komponen `<ion-datetime>` (Kode 2.49), dan di antarmuka pemilih yang ditampilkan dari bawah layar.

```
1 <ion-datetime displayFormat="MM DD YY" placeholder="Select Date"></ion-datetime>
```

Kode 2.36: Kode Program dari Datetime Component dengan Format Bulan-Hari-Tahun

- Grid

Grid digunakan untuk membuat tata letak kustom pada tampilan. Grid terdiri dari tiga bagian, yaitu *grid*, baris, dan kolom. Masing-masing kolom dapat diubah ukurannya menggunakan CSS. Grid digunakan dengan tag `<ion-grid>` (Kode 2.37).

```
1 <ion-row>
2   <ion-col size="6">
3     ion-col [size="6"]
4   </ion-col>
5   <ion-col>
6     ion-col
7   </ion-col>
8 </ion-row>
```

Kode 2.37: Potongan Kode Program dari Grid Component

- Infinite Scroll

Komponen Infinite Scroll memanggil sebuah action yang akan dilakukan ketika pengguna menggulir dengan jarak tertentu dari bawah atau atas halaman. Penggunaan Infinite Scroll dengan menggunakan tag `<ion-infinite-scroll>` (Kode 2.38).

```
1 <ion-infinite-scroll threshold="100px" (ionInfinite)="loadData($event)">
2   <ion-infinite-scroll-content
3     loadingSpinner="bubbles"
4     loadingText="Loading more data...">
5   </ion-infinite-scroll-content>
6 </ion-infinite-scroll>
```

Kode 2.38: Potongan Kode Program dari Infinite Scroll Component

- Icon

Icon merupakan komponen yang berupa gambar kecil, yang merepresentasikan sebuah berkas, dan folder di dalam aplikasi. Penggunaan Icon adalah dengan menggunakan tag `<ion-icon>` (Kode 2.39).

```
1 <ion-icon name="home"></ion-icon>
```

Kode 2.39: Potongan Kode Program dari Icon Home

- Item

Item merupakan elemen yang dapat berisi teks, ikon, avatar, gambar, masukan, dan elemen asli atau kustom lainnya. Biasanya, item ditempatkan di dalam sebuah *list* bersamaan dengan item lainnya dengan tag `<ion-item>` (Kode 2.40). Dapat dilakukan *swipe*, dihapus, disusun ulang, dan diedit.

```
1 <ion-item>
2   <ion-label>
3     Item
4   </ion-label>
5 </ion-item>
```

Kode 2.40: Potongan Kode Program dari Item Component

- List

Komponen List terdiri dari beberapa baris *item* yang dapat berisi teks, tombol, *toggles*, ikon, thumbnails, dan komponen-komponen lainnya menggunakan tag <ion-list> (Kode 2.41). List mendukung untuk pengguna melakukan *swiping*, *dragging*, dan penghapusan *item*.

```

1 <ion-list>
2   <ion-item>
3     <ion-label>Pokemon Yellow</ion-label>
4   </ion-item>
5   <ion-item>
6     <ion-label>Mega Man X</ion-label>
7   </ion-item>
8   <ion-item>
9     <ion-label>The Legend of Zelda</ion-label>
10  </ion-item>
11  <ion-item>
12    <ion-label>Pac-Man</ion-label>
13  </ion-item>
14  <ion-item>
15    <ion-label>Super Mario World</ion-label>
16  </ion-item>
17 </ion-list>
```

Kode 2.41: Potongan Kode Program dari List Component

- Menu

Komponen Menu merupakan panel navigasi samping yang dapat dilakukan *slides* dari sisi pada tampilan halaman saat ini menggunakan tag <ion-menu> (Kode 2.42). Pada dasarnya, Menu muncul dari kiri, tetapi sisi kemunculan menu dapat diganti.

```

1 <ion-menu side="start" menuId="first" contentId="main">
2   <ion-header>
3     <ion-toolbar color="primary">
4       <ion-title>Start Menu</ion-title>
5     </ion-toolbar>
6   </ion-header>
7   <ion-content>
8     <ion-list>
9       <ion-item>Menu Item</ion-item>
10      <ion-item>Menu Item</ion-item>
11      <ion-item>Menu Item</ion-item>
12      <ion-item>Menu Item</ion-item>
13      <ion-item>Menu Item</ion-item>
14     </ion-list>
15   </ion-content>
16 </ion-menu>
```

Kode 2.42: Potongan Kode Program dari Menu Component

- Modal

Modal merupakan kotak dialog yang muncul diatas konten aplikasi lain, dan harus diutup secara manual oleh pengguna sebelum pengguna dapat melanjutkan menggunakan aplikasi. Modal berguna sebagai komponen pilihan ketika ada banyak opsi untuk dipilih, atau melakukan penyaringan isi di dalam daftar, serta beberapa kasus serupa lainnya. Modal dapat digunakan dengan tag <ion-modal> (Kode 2.43).

```

1 <ion-modal [isOpen]="true">
2   <ng-template>
3     <ion-content>Modal Content</ion-content>
4   </ng-template>
5 </ion-modal>
```

Kode 2.43: Kode Program dari Modal

- Navigation

Navigation adalah komponen mandiri yang digunakan untuk membuat komponen baru ke dalam *stack*. Navigation tidak terikat kepada *router* tertentu, mengakibatkan jika kita membuat komponen Navigation dan melakukan *push* komponen lain ke dalam *stack*, komponen tersebut tidak akan mempengaruhi *router* aplikasi secara keseluruhan. Sesuai dengan kasus penggunaan dimana ketika pengguna bisa memilih modal, yang membutuhkan sub-navigasinya sendiri, tanpa membuatnya terikat ke URL aplikasi.

- Refresher

Refresher merupakan komponen yang menyediakan fungsionalitas *pull-to-refresh* pada komponen konten dengan tag `<ion-refresher>` (Kode 2.44). Refresher digunakan pada saat pengguna menarik layar dari atas ke bawah, maka Refresher akan menyegarkan data atau mendapatkan daftar data untuk mengambil lebih banyak data.

```

1 <ion-content>
2   <ion-refresher slot="fixed" (ionRefresh)="doRefresh($event)">
3     <ion-refresher-content></ion-refresher-content>
4   </ion-refresher>
5 </ion-content>

```

Kode 2.44: Kode Program dari Refresher

- Segment

Segment berfungsi untuk menampilkan pilihan tombol bagi pengguna untuk beralih di antara tampilan berbeda di dalam satu halaman yang sama. Segment menampilkan sekelompok tombol-tombol yang dapat diklik, dalam baris horizontal. Penggunaan Segment dengan menggunakan tag `<ion-segment>` (Kode 2.45).

```

1 <ion-segment (ionChange)="segmentChanged($event)">
2   <ion-segment-button value="friends">
3     <ion-label>Friends</ion-label>
4   </ion-segment-button>
5   <ion-segment-button value="enemies">
6     <ion-label>Enemies</ion-label>
7   </ion-segment-button>
8 </ion-segment>

```

Kode 2.45: Kode Program dari Segment

- Slides

Komponen Slides merupakan sebuah *multi-section container* yang setiap bagianya dapat dilakukan *swipe* atau *drag*. Untuk menggunakan komponen ini dengan menggunakan tag `<ion-slides>` seperti pada kode 2.46 sebagai pembungkus slide. Masing-masing slide menggunakan tag `<ion-slide>` seperti pada baris ke-2. Tag tersebut dapat digunakan untuk menggeser slide dari kiri ke kanan, atau sebaliknya.

```

1 <ion-slides pager="true" [options]="slideOpts">
2   <ion-slide>
3     <h1>Slide 1</h1>
4   </ion-slide>
5   <ion-slide>
6     <h1>Slide 2</h1>
7   </ion-slide>
8   <ion-slide>
9     <h1>Slide 3</h1>
10  </ion-slide>
11 </ion-slides>

```

Kode 2.46: Kode Program dari Slides

- Tabs

Tabs merupakan navigasi *top-level* yang mengimplementasi sebuah *tab-based navigation*. Tabs dapat digunakan dengan tag `<ion-tabs>` (Kode 2.47) yang tidak memiliki *styling* apapun dan bekerja sebagai *router outlet* untuk menangani navigasi.

```

1 <ion-tabs>
2   <ion-tab-bar slot="bottom">
3     <ion-tab-button tab="schedule">
4       <ion-icon name="calendar"></ion-icon>
5       <ion-label>Schedule</ion-label>
6       <ion-badge>6</ion-badge>
7     </ion-tab-button>
8
9     <ion-tab-button tab="speakers">
10      <ion-icon name="person-circle"></ion-icon>
11      <ion-label>Speakers</ion-label>
12    </ion-tab-button>
13  </ion-tab-bar>
14 </ion-tabs>

```

Kode 2.47: Kode Program dari Tabs

- Toast

Komponen Toast merupakan sebuah notifikasi yang digunakan di aplikasi modern untuk memberikan umpan balik atau menampilkan pesan sistem. Komponen ini muncul diatas konten aplikasi, dan dapat ditutup untuk melanjutkan penggunaan aplikasi. Komponen ini menggunakan `ToastController` yang dimiliki oleh *library* Ionic Angular (Kode 2.48)

```

1 async presentToast() {
2   const toast = await this.toastController.create({
3     message: 'Your settings have been saved.',
4     duration: 2000
5   });
6   toast.present();
7 }

```

Kode 2.48: Kode Program dari Toast

- Toolbar

Toolbar dapat diposisikan di atas ataupun di bawah konten. Ketika toolbar ditempatkan di header `<ion-header>` akan muncul di bagian atas konten, sedangkan ketika ditempatkan di footer `<ion-footer>` akan muncul tetap di bagian bawah. Toolbar menggunakan tag `<ion-toolbar>`, yang di dalamnya dapat berisi button, dan dapat menggunakan border (Kode 2.49).

```

1 <ion-toolbar>
2   <ion-buttons slot="start">
3     <ion-back-button></ion-back-button>
4   </ion-buttons>
5   <ion-title>Back Button</ion-title>
6 </ion-toolbar>

```

Kode 2.49: Kode Program dari Toolbar dengan Button di Dalamnya

Selain komponen-komponen yang telah disebutkan, tertapat beberapa komponen lainnya yang tidak disebutkan disini. Komponen-komponen tersebut yaitu Checkbox, Chip, Floating Action Button, Grid, Input, Popover, Progress Indicator, Radio, Reorder, Routing, Searchbar, Select, dan Toggle ¹.

¹ ‘UI Components’ <https://ionicframework.com/docs/components>, Diakses pada 17 April 2022.

2.3.3 Migrasi Ionic 3 ke Ionic 6

Untuk melakukan migrasi dari Ionic 3 ke Ionic 6 memerlukan tiga tahap, yaitu migrasi dari Ionic 3 ke Ionic 4, migrasi Ionic 4 ke Ionic 5, dan migrasi dari Ionic 5 ke Ionic 6. Tahapan migrasi tersebut adalah sebagai berikut:

1. Migrasi Ionic 3 ke Ionic 4

Ada beberapa langkah untuk melakukan migrasi dari Ionic 3 ke dalam Ionic 4, yaitu:

- (a) Membuat Proyek Ionic Baru

Untuk membuat projek Ionic baru tanpa *template* apapun dengan menggunakan perintah **ionic start myApp blank** dan memilih Angular sebagai *frameworknya* [2.50](#).

```
1 ionic start myApp blank
```

Kode 2.50: Perintah Membuat Proyek Ionic Baru

- (b) Menyalin Angular Services yang pada Ionic 3 berada di **src/providers**, menjadi **src/app/services** pada Ionic 4.

- (c) Menyalin *Root-level Items*

Menyalin seluruh *Root-level Items* pada Ionic versi 3, seperti *pipes* dan *components*. Terdapat perubahan struktur direktori dengan perubahan direktori yang semula **src/components** pada Ionic 3, menjadi **src/app/components** pada Ionic 4.

- (d) Menyalin Global Scss dari **src/app/app.scss** pada Ionic 3, menjadi **src/global.scss** pada Ionic 4.

- (e) Menyalin Bagian-bagian Aplikasi

Menyalin keseluruhan bagian yang ada pada aplikasi, baik itu halaman maupun fitur yang ada, dengan ketentuan sebagai berikut :

- Shadow DOM sudah aktif secara *default*.
- Page atau Components Sass tidak lagi dibungkus dengan *tag page* atau *components* dan harus menggunakan opsi styleUrls milik Angular dari dekorator @Component.
- Perubahan RxJS yang digunakan. Pada ionic 3 adalah versi 5, sedangkan pada Ionic 4 RxJS yang digunakan adalah versi 6.
- Lifecycle Hooks tertentu harus digantikan dengan Angular Hooks.
- Perubahan markup yang mungkin saja dibutuhkan.

Sejak Ionic 4 dipindahkan ke elemen kustom, terdapat perubahan yang signifikan terkait dengan markup untuk setiap komponen. Semua perubahan ini dibuat untuk mengikuti spesifikasi dari elemen kustom. Komponen-komponen yang berubah tersebut yaitu :

- *Button*

Terdapat perbedaan pada *tag* untuk membuat Button, yang semula pada Ionic 3 adalah **<button>** menjadi **<ion-button>** pada Ionic 4 (Kode [2.51](#)).

```
1 <ion-button (click)="doSomething()">
2   Default Button
3 </ion-button>
```

Kode 2.51: Penggunaan Button pada Ionic 4

- *Floating Action Button (FAB)*

Terdapat perbedaan pada *tag* di dalam **<ion-fab>**, yang semula pada Ionic 3 adalah **<button>** menjadi **<ion-fab-button>** pada Ionic 4 (Kode [2.52](#)).

```
1 <ion-fab>
2   <ion-fab-button>
3     <ion-icon name="add"></ion-icon>
4   </ion-fab-button>
5   <ion-fab-list>
6     <ion-fab-button>
7       <ion-icon name="logo-facebook"></ion-icon>
8     </ion-fab-button>
9   </ion-fab-list>
```

10 | </ion-fab>

Kode 2.52: Penggunaan Floating Action Button pada Ionic 4

- Item

Terdapat perbedaan pada tag `<ion-item>`, dimana pada Ionic 3, tag `<ion-label>` akan secara otomatis ditambahkan ke dalam `<ion-item>`. Sedangkan pada Ionic 4 diharuskan untuk menambahkan tag `<ion-label>` secara manual ke dalam komponen item (Kode 2.53).

```
1 <ion-item>
2   <ion-label>
3     Default Item
4   </ion-label>
5 </ion-item>
```

Kode 2.53: Penggunaan Item pada Ionic 4

- Label

Pada Ionic 4, atribut untuk mengatur posisi dari label digabungkan dengan atribut *position* (Kode 2.54).

```
1 <ion-item>
2   <ion-label position="floating">Floating Label</ion-label>
3   <!-- input -->
4 </ion-item>
```

Kode 2.54: Penggunaan Atribut *Position* pada Ionic 4

- Menu

Terdapat beberapa perubahan nama pada Ionic 4, yaitu:

- * Perubahan Nama Properti Terdapat perubahan nama properti pada Ionic 4. Perubahan-perubahan tersebut adalah sebagai berikut:

Properti	Perubahan	
	Ionic 3	Ionic 4
swipeEnabled	swipeEnabled	swipeGesture
content	content	contentId

- * Perubahan Nama Events Terdapat perubahan nama *events* pada Ionic 4. Perubahan-perubahan tersebut adalah sebagai berikut :

Events	Perubahan	
	Ionic 3	Ionic 4
ionClose	ionClose	ionDidClose
ionOpen	ionOpen	ionDidOpen

- Nav

Terdapat perubahan Nav pada Ionic 4. Perubahan-perubahan tersebut adalah sebagai berikut:

- * Perubahan Nama Method Terdapat perubahan nama *method* pada Ionic 4. Perubahan-perubahan tersebut adalah sebagai berikut:

Nama Method	Perubahan	
	Ionic 3	Ionic 4
remove	remove	getChildNavs
getActiveChildNavs	getActiveChildNavs	getChildNavs

- * Perubahan Nama Prop

Terdapat perubahan nama prop pada Ionic 4. Perubahan tersebut adalah sebagai berikut:

Nama Prop	Perubahan	
	Ionic 3	Ionic 4
swipeBackEnabled	swipeBackEnabled	swipeGesture

- Navbar

Pada Ionic 4, terdapat penghapusan terhadap komponen `<ion-navbar>` karena untuk menjaga agar selalu menggunakan `<ion-toolbar>` dengan *back button* yang eksplisit (Kode 2.55).

```

1 <ion-toolbar>
2   <ion-buttons slot="start">
3     <ion-back-button></ion-back-button>
4   </ion-buttons>
5   <ion-title>My Navigation Bar</ion-title>
6 </ion-toolbar>

```

Kode 2.55: Penggunaan Navbar pada Ionic 4 dengan *Back Button*

- Overlays

Pada Ionic 4, semua overlay harus menggunakan `async/await`. Overlay tersebut adalah Action Sheet, Alert, Loading, Modal, Popover, and Toast (Kode 2.56). Terjadi perubahan nama properti `enableBackdropDismiss` menjadi `backdropDismiss`.

```

1 async presentToast() {
2   const toast = await this.toastController.create({
3     message: 'Your settings have been saved.',
4     duration: 2000
5   });
6   toast.present();
7 }

```

Kode 2.56: Penggunaan Overlay untuk Toast pada Ionic 4

- Scroll

Tag `<ion-scroll>` sudah dihapus pada Ionic 4 agar penggunaan tag `<ion-content>` menjadi lebih maksimal.

- Segment Button

Pada Ionic 4, teks pada Segment Button membutuhkan `<ion-label>` untuk membungkusnya (Kode 2.57).

```

1 <ion-segment-button>
2   <ion-label>Item One</ion-label>
3 </ion-segment-button>

```

Kode 2.57: Penggunaan Segment Button untuk Toast pada Ionic 4

Selain yang telah disebutkan, terdapat beberapa perubahan lainnya yang tidak ditulis seperti Action Sheet, Alert, Colors, Content, Datetime, Dynamic Mode, Fixed Content, Grid, Icon, Infinite Scroll, Item Divider, Item Options, Item Sliding, List Header, Loading, Modal, Option, Popover, Radio, Range, Refresher, Select, Show When, Hide When, Spinner, Tabs, Typography, Theming, dan Toolbar ².

² ‘Breaking Changes’ <https://github.com/ionic-team/ionic-framework/blob/main/angular/BREAKING.md>, Diakses pada 13 November 2021.

2. Migrasi Ionic 4 ke Ionic 5

Migrasi aplikasi dari Ionic 4 ke Ionic 5 memerlukan beberapa pembaruan mengenai properti API, CSS, dan *package dependencies* yang terpasang. Perubahan-perubahan tersebut yaitu :

- CSS

- *Activated, Focused, Hover States*

Kelas `.activated` secara otomatis ditambahkan ke komponen yang dapat diklik, mengalami perubahan nama menjadi `.ion-activated`. Terdapat pembaruan komponen Action Sheet sehingga variabel akan diawali dengan `button`. Hal ini dapat memungkinkan aplikasi tetap memiliki kontrol atas `opacity` jika diinginkan, tetapi saat memperbarui status, hanya perlu mengatur variabel utama, yaitu `-background-activated`, `-background-focused`, `-background-hover`. Hal tersebut penting saat mengubah tema global, karena memperbarui warna `toolbar` akan secara otomatis memperbarui *hover states* untuk semua `buttons` di `toolbar` (Kode 2.58).

```

1 /* Setting the button background on hover to solid red */
2 ion-button {
3   --background-hover: red;
4   --background-hover-opacity: 1;
5 }
6
7 /* Setting the action sheet button background on focus to an opaque green
8  */
9 ion-action-sheet {
10   --button-background-focus: green;
11   --button-background-focus-opacity: 0.5;
12 }
13 /*
14 * Setting the fab button background on hover to match the text color with
15 * the default --background-hover-opacity on md
16 */
17 .md ion-fab-button {
18   --color: #222;
19   --background-hover: #222;
20 }
```

Kode 2.58: Contoh Kode *Hover States* pada Ionic 5

- *CSS Utilities*

Karena pada versi sebelumnya, yaitu Ionic versi 4, terdapat masalah dengan menggunakan atribut CSS dengan *framework* yang menggunakan JSX dan TypeScript, Ionic *Framework* menambahkan dukungan untuk beberapa *framework*, dan pada Ionic 5 menambahkan kelas CSS. Ionic versi 5 menghapus atribut CSS dan mendukung konsistensi. Ionic versi 5 juga mengubah ke kelas dengan diawali `ion` untuk menghindari konflik dengan atribut asli dan CSS dari pengguna (Kode 2.59).

```

1 <ion-header class="ion-text-center"></ion-header>
2 <ion-content class="ion-padding"></ion-content>
3 <ion-label class="ion-text-wrap"></ion-label>
4 <ion-item class="ion-wrap"></ion-item>
```

Kode 2.59: Contoh Kode Kelas CSS *Utility* pada Ionic 5

- *Display Classes*

Kelas dari *responsive display* yang ditemukan di dalam berkas `display.css` memiliki kueri media yang diperbarui untuk lebih mencerminkan bagaimana cara kerjanya.

- *Distributed Scss*

Berkas `scss` telah dihapus dari `dist/`. Sebagai gantinya, variabel CSS harus digunakan untuk tema.

- Komponen

Terdapat perubahan beberapa komponen pada Ionic 5, yaitu :

- Back Button dan Button

Perubahan terdapat pada penambahan penamaan kelas `.activated` yang secara otomatis ditambahkan ke komponen yang dapat di klik, menjadi `.ion-activated`.

- Controllers

Terdapat beberapa komponen yang dihapus dari Ionic sebagai elemen, yaitu `ion-action-sheet-controller`, `ion-alert-controller`, `ion-loading-controller`, `ion-menu-controller`, `ion-modal-controller`, `ion-picker-controller`, `ion-popover-controller`, dan `ion-toast-controller`. Sebagai gantinya, maka harus diimpor dari `@ionic/core`.

- Header dan Footer

Atribut `no-border` dihapus, dan sebagai gantinya yaitu dengan menggunakan kelas `ion-no-border`.

- List Header

Konten berupa teks apa pun di dalam `<ion-list-header>` harus dibungkus dengan `<ion-label>` sesuai dengan gaya desain yang baru (Kode 2.60). Jika label tidak ada, maka perataan tombol di header bisa saja terlihat tidak aktif.

```

1 <ion-list-header>
2   <ion-label>New This Week</ion-label>
3   <ion-button>See All</ion-button>
4 </ion-list-header>

```

Kode 2.60: Kode Program untuk List Header

- Menu

Fungsi `swipeEnable()` telah dihapus di Angular, sebagai gantinya menggunakan `swipeGesture()`. Nilai `left` dan `right` telah dihapus, dan menggunakan `start` dan `end` sebagai gantinya. Terdapat penghapusan atribut utama, sebagai gantinya yaitu dengan menggunakan `content-id` (untuk vanila JS atau Vue) dan `contentId` (untuk Angular atau React) (Kode 2.61).

```

1 <ion-menu content-id="main"></ion-menu>
2 <ion-content id="main">...</ion-content>

```

Kode 2.61: Kode Program untuk Menu

- Select Option

Properti `selected` telah dihapus. Sebagai gantinya harus mengatur properti nilai pada `ion-select` induk agar sesuai dengan opsi terpilih yang diinginkan (Kode 2.62).

```

1 <ion-select value="two">
2   <ion-select-option value="one">One</ion-select-option>
3   <ion-select-option value="two">Two</ion-select-option>
4 </ion-select>

```

Kode 2.62: Kode Program untuk Select Option

- Toast

Properti `close button` seperti `showCloseButton` dan `closeButtonText` telah dihapus. Sebagai gantinya, gunakan `buttons` array untuk fungsi batal (Kode 2.63).

```

1 async presentToast() {
2   const toast = await this.toastController.create({
3     message: 'Your settings have been saved.',
4     buttons: [
5       {
6         text: 'Close',
7         role: 'cancel',
8         handler: () => {

```

```

9     console.log('Close clicked');
10    }
11    }
12  ]
13 });
14 toast.present();
15 }

```

Kode 2.63: Kode Program untuk Toast

Selain yang sudah disebutkan, terdapat beberapa komponen lain yang mendapat perubahan di Ionic 5, namun tidak dituliskan dalam dokumen skripsi ini. Komponen-komponen tersebut antara lain Action Sheet, Anchor, Card, FAB, Item, Menu Button, Nav Link, Radio, Segment, Segment Button, Skeleton Text, Split Pane, dan Tabs³.

- *Package dan Dependencies*

Untuk memasang *package* dan *dependencies* pada Angular, dapat memanfaatkan npm pada CLI, dengan menjalankan pemasangan pada *package ionic-angular* (Kode 2.64). Namun jika ingin membuat proyek baru, dapat dibuat dari CLI dan aplikasi yang ada dapat dimigrasikan secara manual.

```
1 npm install @ionic/angular@latest @ionic/angular-toolkit@latest --save
```

Kode 2.64: Kode untuk Memasang *Package* dan *Dependencies* pada Angular

- Warna

Terdapat perubahan terhadap warna bawaan milik ionic (Tabel 2.1).

Tabel 2.1: Tabel Warna Bawaan di Ionic 5

Nama Warna	Kode HEX
primary	#3880ff
secondary	#3dc2ff
tertiary	#5260ff
success	#2dd36f
warning	#ffc409
danger	#eb445a
light	#f4f5f8
medium	#92949c
dark	#222428

- Events

Pada Ionic 5, Events services di @ionic/angular telah dihapus. Sebagai gantinya gunakan Observables untuk arsitektur pub/sub, dan Redux untuk *advanced state management*.

3. Migrasi Ionic 5 ke Ionic 6

Berikut merupakan perubahan-perubahan pada Ionic 6, diantaranya yaitu:

- Pembaruan Ionic dan Angular

Ionic 6 mendukung penggunaan Angular versi 12 dan yang lebih baru, dengan begitu perlu dilakukan perbaruan Angular ke versi yang terbaru (Kode 2.65).

```
1 npm install @ionic/angular@6
```

Kode 2.65: Kode untuk Memperbarui Versi Ionic 6 dengan versi Angular Terbaru

- Mengganti penggunaan *Config.set()* menjadi *IonicModule.forRoot()*.

³ ‘Breaking Changes’ <https://github.com/ionic-team/ionic-framework/blob/main/BREAKING.md>, Diakses pada 20 November 2021.

- *Icon*

Penghapusan properti ariaLabel dan ariaHidden, dan diganti dengan menggunakan aria-label dan aria-hidden.

- *Input, Select, dan Textarea*

Menggunakan “undefined” sebagai nilai untuk diteruskan ke porperti placeholder, dibandingkan dengan menggunakan “null”.

- *Modal dan Popover*

ion-modal (Kode 2.66) dan ion-popover (Kode 2.67) menggunakan Shadow DOM.

```
1 ion-modal::part(content) {  
2   ...  
3 }  
4  
5 ion-modal::part(backdrop) {  
6   ...  
7 }
```

Kode 2.66: Kode ion-modal menggunakan Shadow DOM pada CSS

```
1 ion-popover::part(arrow) {  
2   ...  
3 }  
4  
5 ion-popover::part(backdrop) {  
6   ...  
7 }  
8  
9 ion-popover::part(content) {  
10  ...  
11 }
```

Kode 2.67: Kode ion-popover menggunakan Shadow DOM pada CSS

- *Radio*

Menghapus semua penggunaan antarmuka RadioChangeEventDetail.

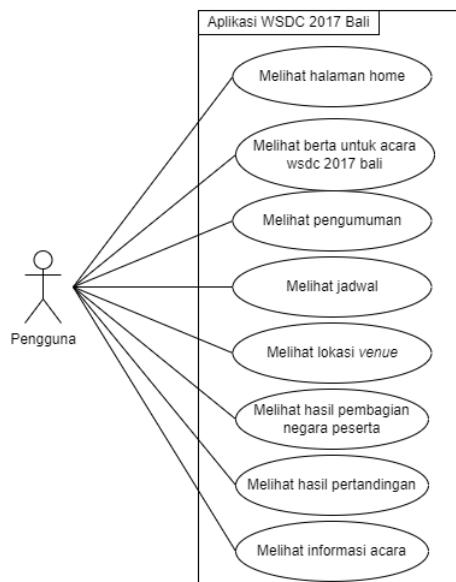
BAB 3

ANALISIS

Pada bab ini menjelaskan analisis aplikasi WSDC 2017 Bali saat ini dan aplikasi WSDC yang akan dibangun, dan tantangan pada pengembangan sistem usulan. Analisis yang akan dibahas meliputi analisis *use case*, analisis kebutuhan sistem, dan analisis pembangunan aplikasi Android WSDC 2017 Bali menggunakan Ionic.

3.1 Analisis Sistem Kini dan Sistem Usulan

Aplikasi WSDC 2017 Bali digunakan untuk menunjang keberlangsungan acara WSDC 2017 yang diselenggarakan di Bali, Indonesia. Pada halaman utama, pengguna dapat melihat berita-berita terkait acara WSDC 2017 Bali dan tombol *read more* yang apabila ditekan akan mengarahkan pengguna untuk melihat berita terkait acara WSDC 2017 Bali dengan format pdf. Aplikasi WSDC 2017 Bali dapat digunakan untuk melihat berita acara, pengumuman, jadwal peserta, lokasi acara, hasil pengundian, info, serta pengumuman pemenang dari acara WSDC 2017 Bali (Gambar 3.1).



Gambar 3.1: *Use Case Diagram* Aplikasi WSDC 2017 Bali

Terdapat *sidemenu* untuk pengguna agar dapat bernavigasi ke dalam menu-menu yang terdapat pada aplikasi WSDC 2017 Bali. Untuk mengakses *sidemenu*, pengguna dapat menekan tombol navigasi berada di sebelah kiri atas aplikasi WSDC 2017 Bali. Selain cara tersebut dapat juga dengan cara mengusap layar dari kiri ke kanan. Untuk menutup *sidemenu*, pengguna dapat menekan area di luar *sidemenu*, atau dengan cara menekan tombol silang di sebelah kiri atas *sidemenu*. Terdapat fitur-fitur yang ada pada aplikasi WSDC 2017 Bali yang dapat diakses melalui *sidemenu*. Fitur-fitur tersebut adalah sebagai berikut :

1. Home

Pada halaman ini, pengguna dapat melihat halaman utama aplikasi WSDC 2017 Bali yang berisi berita acara WSDC 2017 Bali, serta pemberitahuan terakhir terkait acara WSDC 2017 Bali. Halaman ini merupakan halaman awal yang ditampilkan saat aplikasi WSDC 2017 Bali pertama kali dibuka (Tabel 3.1). Untuk mengakses halaman ini, dapat melalui *sidemenu*.

Tabel 3.1: Tabel Skenario dari Halaman *Home*

No	Aksi Aktor	Reaksi Sistem
1	Pengguna membuka aplikasi WSDC 2017 Bali	Aplikasi WSDC 2017 Bali menampilkan halaman selamat datang.
2		Aplikasi WSDC 2017 Bali menampilkan halaman <i>Home</i>
3	Pengguna mengklik <i>card Announcements</i>	Aplikasi WSDC 2017 Bali menampilkan halaman <i>Announcement</i> .

2. Newsletter

Pada *bagian newsletter* yang terdapat di *home*, pengguna dapat melihat berita-berita terkait acara WSDC 2017 Bali dengan format pdf. Untuk mengakses halaman ini, dapat melalui *sidemenu* (Tabel 3.2).

Tabel 3.2: Tabel Skenario dari *Newsletter*

No	Aksi Aktor	Reaksi Sistem
1	Pengguna menekan tombol <i>read more</i> pada berita di halaman utama aplikasi WSDC 2017 Bali.	Aplikasi WSDC 2017 Bali menampilkan berita pada acara WSDC 2017 Bali

3. Announcement

Pengguna dapat melihat berbagai pengumuman mengenai keberlangsungan acara WSDC 2017 Bali yang tersusun berdasarkan tanggal dirilisnya pengumuman tersebut. Untuk mengakses halaman ini, dapat melalui *sidemenu* (Tabel 3.3).

Tabel 3.3: Tabel Skenario dari Halaman *Announcement*

No	Aksi Aktor	Reaksi Sistem
1	Pengguna menekan tombol <i>Announcement</i> pada <i>sidemenu</i>	Aplikasi WSDC 2017 Bali menampilkan halaman <i>Announcement</i> .

4. Schedule

Pada halaman ini, pengguna dapat melihat jadwal acara WSDC 2017 Bali yang ditampilkan berkelompok berdasarkan tanggal dan hari. Jadwal yang ditampilkan berupa waktu mulai dan waktu selesai, lokasi acara, serta nama acara. Untuk mengakses halaman ini, dapat melalui *sidemenu* (Tabel 3.4).

Tabel 3.4: Tabel Skenario dari Halaman *Schedule*

No	Aksi Aktor	Reaksi Sistem
1	Pengguna menekan tombol <i>Schedule</i> pada <i>sidemenu</i>	Aplikasi WSDC 2017 Bali menampilkan halaman <i>Schedule</i> .
2	Pengguna menekan tanggal yang berada di atas halaman jadwal	Aplikasi WSDC 2017 Bali menampilkan jadwal berdasarkan tanggal yang dipilih oleh pengguna dengan detail waktu, lokasi, dan nama kegiatan.

5. Venues

Pada halaman ini, pengguna dapat melihat lokasi dari berlangsungnya acara WSDC 2017 Bali. Untuk mengakses halaman ini, dapat melalui sidemenu (Tabel 3.5).

Tabel 3.5: Tabel Skenario dari Halaman *Venues*

No	Aksi Aktor	Reaksi Sistem
1	Pengguna menekan tombol <i>Venues</i> pada <i>sidemenu</i>	Aplikasi WSDC 2017 Bali menampilkan halaman <i>Venues</i> yang berisi <i>Ceremony Venues</i> , <i>Competition Venues</i> , <i>Delegates Accomodation</i> , dan <i>Educational Tour</i> .
2	Pengguna menekan kategori <i>venues</i> yang diinginkan.	Aplikasi WSDC 2017 Bali menampilkan peta, nama lokasi acara dengan disertai penanda yang ada di dalam peta, dan jarak antara lokasi pengguna saat ini dan lokasi acara.

6. Draw

Pada halaman ini, pengguna dapat melihat pembagian *venue* serta pembagian kubu proposisi dan oposisi dari hasil pengundian untuk para negara peserta WSDC 2017 Bali. Untuk mengakses halaman ini, dapat melalui *sidemenu* (Tabel 3.6).

Tabel 3.6: Tabel Skenario dari Halaman *Draw*

No	Aksi Aktor	Reaksi Sistem
1	Pengguna menekan tombol <i>Draw</i> pada <i>sidemenu</i>	Aplikasi WSDC 2017 Bali menampilkan halaman <i>Draw</i> yang dapat digulir kebawah untuk menampilkan keseluruhan tabel.

7. Result

Pada halaman ini, pengguna dapat melihat pemenang dari kompetisi WSDC 2017 Bali. Untuk mengakses halaman ini, dapat melalui *sidemenu* (Tabel 3.7).

Tabel 3.7: Tabel Skenario dari Halaman *Result*

No	Aksi Aktor	Reaksi Sistem
1	Pengguna menekan tombol <i>Result</i> pada <i>sidemenu</i>	Aplikasi WSDC 2017 Bali menampilkan halaman <i>Result</i> yang berisi pemenang dari babak semifinal, seperempatfinal, dan seperdelapan-final.

8. Info

Pada halaman ini, pengguna dapat melihat info-info seputar kontak-kontak penting yang dapat dihubungi, kosa kata dalam Bahasa Indonesia sehari-hari, serta *credits* kepada pembuat aplikasi WSDC 2017 Bali. Untuk mengakses halaman ini, dapat melalui *sidemenu* (Tabel 3.8).

Tabel 3.8: Tabel Skenario dari Halaman Info

No	Aksi Aktor	Reaksi Sistem
1	Pengguna menekan tombol <i>hamburger</i> di pojok kiri atas atau melakukan <i>swipe</i> dari kiri layar ke kanan layar aplikasi WSDC 2017 Bali.	Aplikasi WSDC 2017 Bali menampilkan <i>sidebar</i>
2	Pengguna menekan tombol Info	Aplikasi WSDC 2017 Bali menampilkan halaman Info

Aplikasi WSDC 2017 Bali saat ini menggunakan Ionic versi 3, Angular versi 4.0.0, dan Cordova. Dengan Ionic Framework yang disusun berdasarkan arsitektur Angular, maka aplikasi WSDC 2017 memungkinkan untuk ditulis menggunakan bahasa pemrograman web seperti HTML, CSS, dan Javascript. Pada Ionic Framework versi 3 juga terdapat UI Component [2.3.2](#) yang digunakan dalam aplikasi WSDC 2017 Bali, diantarnya yaitu Badge, Button, Card, Content, Grid, Icons, Items, List, Menu, Segment, Slides, Tabs, dan Toolbar. Kemudian dengan digunakannya Cordova, maka seluruh kode program yang menggunakan bahasa pemrograman web tersebut, dapat hidup dan berjalan seperti halnya aplikasi *native* di dalam perangkat seluler.

Anatomi pada Ionic Framework memiliki struktur proyek Cordova. Pada saat pertama kali dijalankan, aplikasi WSDC 2017 Bali secara *default* akan membuka file index.html yang berada di folder src/index.html. File ini merupakan file pertama yang dijalankan untuk aplikasi WSDC 2017 Bali. Tujuan dari file ini adalah untuk melakukan pengaturan terhadap script, CSS, serta menjalankan aplikasi. Di dalam file index.html ini terdapat sebuah tag <ion-app>. Tag ini yang pertama dicari dan dijalankan oleh Ionic untuk membuka komponen *root* dari aplikasi WSDC 2017 Bali. Pada saat pertama menjalankan aplikasi, kode di dalam folder src akan ditranspilasikan ke versi JavaScript yang dapat dipahami browser. Dengan begitu, aplikasi dapat menjalankan TypeScript yang dikompilasi ke bentuk JavaScript.

Setelah index.html dijalankan, titik masuk ke dalam aplikasi WSDC 2017 Bali adalah file app.module.ts yang berada di src/app/app.module.ts. Di dalam file ini terdapat NgModule untuk mendeklarasi komponen apa saja yang akan digunakan, mengimpor module, bootstrap apa yang digunakan, dan menyediakan services apa yang akan digunakan oleh komponen ([Kode 3.1](#)).

```

1 @NgModule({
2   declarations: [
3     MyApp, HomePage, AnnouncementsPage, SchedulePage, VenuesPage, VenuesMapPage,
4     DrawPage, ResultPage, InfoPage
5   ],
6   imports: [
7     BrowserModule, HttpModule, IonicModule.forRoot(MyApp), IonicModule.forRoot(),
8       CloudModule.forRoot(cloudSettings)
9   ],
10  bootstrap: [IonicApp],
11  entryComponents: [
12    MyApp, HomePage, AnnouncementsPage, SchedulePage, VenuesPage, VenuesMapPage,
13    DrawPage, ResultPage, InfoPage
14  ],
15  providers: [
16    StatusBar, SplashScreen, InAppBrowser, {provide: ErrorHandler, useClass:
17      IonicErrorHandler}, Geolocation,
18  ]
19 })
20 export class AppModule {}
```

Kode 3.1: NgModule pada app.module.ts

Komponen *root* diatur ke MyApp yang berada di folder src/app/app.component.ts. Karena pada file app.component.ts, *root* telah diatur ke dalam MyApp, maka komponen tersebut menjadi komponen pertama yang dibuka ke dalam aplikasi WSDC 2017 Bali. Di dalam komponen tersebut terdapat templateUrl yang digunakan sebagai template utama dari aplikasi WSDC 2017 Bali, yaitu file app.html ([Kode 3.2](#)). Di dalam

template, terdapat tag `<ion-menu>` yang digunakan untuk menampilkan *sidemenu*, lalu tag `<ion-nav>` sebagai area koten utama, dengan properti `[root] = "rootPage"`. Properti tersebut yang nantinya akan diisi oleh halaman *root* dari aplikasi WSDC 2017 Bali, yaitu Home Page. Variabel *rootPage* telah diatur di file *app.component.ts* secara spesifik mengarah ke *HomePage*, yang akan menjadi halaman pertama yang ditampilkan di nav controller.

```

1 <ion-menu [content]="content">
2   <ion-header>
3     <ion-toolbar>
4       <ion-title>
5         <button menuClose id="menu-close-btn">
6           <ion-icon menu-close ios="ios-close-circle-outline" md="md-close-circle"></
7           ion-icon>
8         </button>
9         <span class="text">Menu</span>
10        </ion-title>
11      </ion-toolbar>
12    </ion-header>
13
14    <ion-content>
15      <ion-list>
16        <button class="title-sidemenu" menuClose ion-item *ngFor="let p of pages" (click
17          )="openPage(p)">
18          <ion-icon [ios]=p.iosicon [md]=p.mdicon></ion-icon>
19          <span class="text">{{p.title}}</span>
20        </button>
21      </ion-list>
22    </ion-content>
23
24  </ion-menu>
25
26  <!-- Disable swipe-to-go-back because it's poor UX to combine STGB with side menus -->
27  <ion-nav [root] = "rootPage" #content swipeBackEnabled="false"></ion-nav>

```

Kode 3.2: Source Code File *app.html*

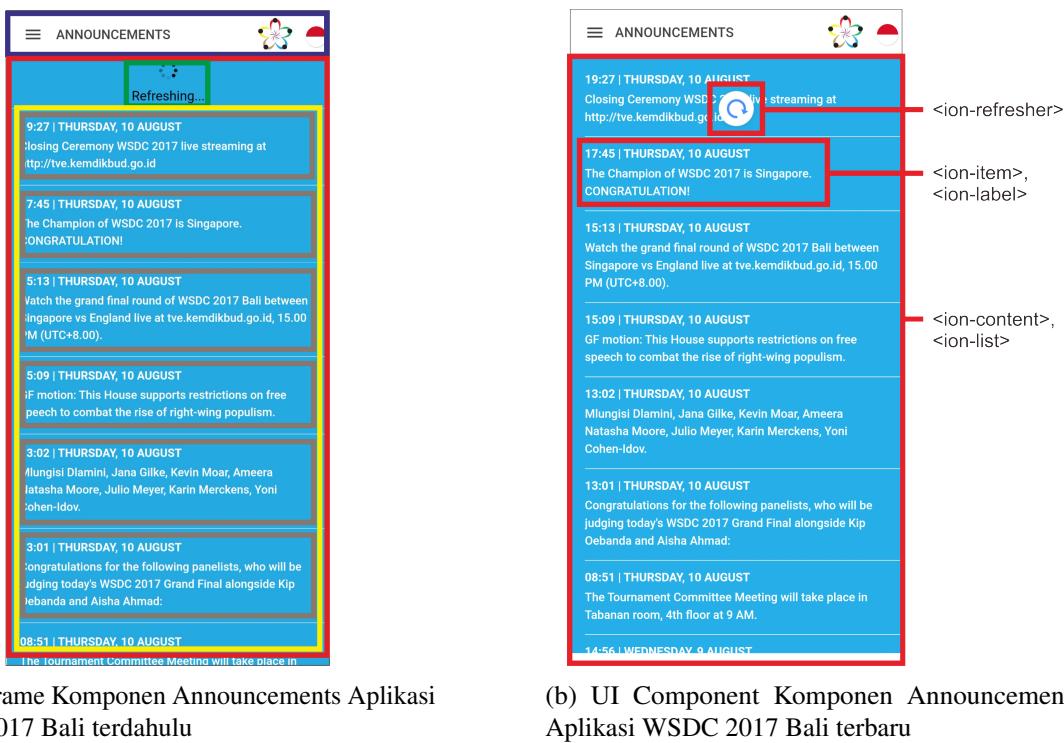
Selain komponen *root*, terdapat beberapa komponen lain yang berisi halaman-halaman yang ada di aplikasi WSDC 2017 Bali. Masing-masing komponen akan mengimpor Component dari `@angular/core`, NavController dari `ionic-angular`, dan Storage dari `@ionic/storage`. Mengimpor Component dari `@angular/core` berfungsi untuk menambahkan sebuah komponen ke dalam *module*. Dengan begitu, komponen tersebut bisa terlihat di seluruh aplikasi, dan dapat digunakan oleh komponen lain. NavController merupakan *base class* untuk mengatur komponen navigasi. Ini berguna agar aplikasi dapat berpindah antar halaman. Sedangkan Storage berfungsi untuk menyimpan pasangan *key/value* dan sebuah objek JSON.

Setiap komponen memiliki tiga buah *file* utama, yaitu *file* HTML, CSS, dan TypeScript. *File* HTML digunakan untuk menampilkan sebuah halaman ke dalam aplikasi dengan susunan kode HTML. *File* CSS digunakan untuk mengatur desain, bentuk, dan tampilan dari sebuah halaman. Sedangkan *file* TypeScript digunakan untuk mengontrol jalannya sebuah komponen.

Komponen-komponen yang ada pada aplikasi WSDC 2017 Bali adalah sebagai berikut:

1. Komponen *Announcement*

Komponen *Announcement* digunakan untuk melihat pengumuman terkait acara WSDC 2017 Bali.



(a) Wireframe Komponen Announcements Aplikasi WSDC 2017 Bali terdahulu

(b) UI Component Komponen Announcements Aplikasi WSDC 2017 Bali terbaru

Gambar 3.2: Komponen Announcements pada Aplikasi WSDC 2017 Bali

(a) Analisis Sistem Kini pada Komponen *Announcements*

Komponen ini digunakan untuk menampilkan halaman *Announcement* pada aplikasi. Komponen ini memiliki sebuah *file* TypeScript untuk mengatur keseluruhan halaman. Di dalam *file* *announcement.ts* terdapat sebuah *decorator* `@Component` untuk komponen (Kode 3.3). Di dalam *decorator* ini terdapat *CSS selector* untuk memilih *CSS* yang akan digunakan, serta **templateUrl** untuk mendefinisikan eksternal *HTML template* yang akan digunakan. *Template HTML* yang digunakan adalah *file* *announcement.html*.

```
1 @Component({
2   selector: 'page-announcements',
3   templateUrl: 'announcements.html',
4 })
```

Kode 3.3: `@Component` pada *announcement.ts*

Pada komponen ini terdapat sebuah kelas *AnnouncementsPage* yang berisi beberapa *method* yang akan digunakan di dalam aplikasi. *Method* pada kelas ini diantaranya adalah sebagai berikut:

- `ionViewDidLoad()`

Method ini merupakan sebuah *lifecycle event* yang dijalankan saat halaman *announcement* telah dimuat. *Event* ini hanya berjalan satu kali, jadi jika sudah melakukan cache terhadap halaman ini, maka `ionViewDidLoad()` tidak akan berjalan lagi. *Lifecycle event* ini berguna untuk melakukan pengaturan awal, yaitu untuk memuat data *announcement* dari *storage*, dan menyimpannya di dalam variabel lokal.

- `doRefresh(refresher)`

Method ini berfungsi untuk melakukan penyegaran ulang pada halaman *announcement* untuk mendapatkan data *announcement* terbaru di dalam server, kemudian menyimpannya ke dalam penyimpanan Ionic. *Method* ini memiliki sebuah parameter *refresher*, yang berisi sebuah *CustomEvent* dari penyegaran ulang yang dilakukan.

- `presentConnectionAlert()`

Method ini digunakan ketika method `doRefresh()` mengalami *error*, yang kemudian memunculkan *toast*.

- `formatDatetime(sqlDatetime: string)`

Method ini berfungsi untuk membuat format tanggal dan waktu. *Method* ini memiliki sebuah parameter, yaitu `sqlDatetime` yang bertipe `string`, yang merupakan sebuah `string` tanggal dengan format “tahun-bulan-hari jam-menit-detik”. *Method* ini akan mengembalikan sebuah teks yang berisi waktu, tanggal dan bulan.

File `announcement.html` digunakan untuk menampilkan halaman *announcemnet*. Terdapat beberapa komponen yang disediakan oleh Ionic Framework, yang diimplementasikan ke dalam halaman *announcement*. Diantaranya adalah sebagai berikut:

- *Header*

Pada *header* dari halaman *announcement*, digunakan beberapa *tag* dari komponen yang disediakan oleh Ionic Framework (Kode 3.4). Yaitu *tag* `<ion-header>` yang merupakan komponen *parent* yang menampung komponen *toolbar* yang ditandai dengan warna biru pada gambar 3.2a. Di dalam *tag* tersebut terdapat *tag* pendukung, seperti `<ion-navbar>`, `<button>` sebagai tombol untuk membuka *sidemenu*, `<ion-icon>` untuk menampilkan icon dari tombol pada *tag button*, dan `<ion-title>` untuk menampilkan judul dari halaman, yaitu *Announcement*, pada *navbar*.

```

1 <ion-header>
2   <ion-navbar>
3     <button ion-button menuToggle>
4       <ion-icon name="menu"></ion-icon>
5     </button>
6     <ion-title>Announcements</ion-title>
7   </ion-navbar>
8 </ion-header>

```

Kode 3.4: *Header* pada Halaman *Announcement*

- *Content*

Konten pada halaman *announcement* yang ditandai dengan kotak berwarna merah pada gambar 3.2a disusun menggunakan *tag* `<ion-content>` (Kode 3.5). *Tag* ini berisi beberapa *tag* lain, yaitu *tag* `<ion-refresher>`, yang ditandai dengan kotak hijau, yang akan menampilkan simbol *refresh* saat pengguna menyegarkan halaman dengan cara melakukan *swipe* dari atas ke bawah layar. Kemudian terdapat *tag* `<ion-list>` yang ditandai dengan kotak kuning, berfungsi untuk menampilkan baris. Baris-baris tersebut diisi menggunakan *tag* `<ion-item>` yang ditandai dengan kotak berwarna hitam, digunakan untuk menyimpan teks yang berisi tanggal, dan pesan pengumuman.

```

1 <ion-content>
2   <ion-refresher (ionRefresh)="doRefresh($event)">
3     <ion-refresher-content pullingIcon="arrow-dropdown" pullingText="Pull
4       to refresh" refreshingSpinner="circles" refreshingText="Refreshing
5       ...">
6     </ion-refresher-content>
7   </ion-refresher>
8   <ion-list>
9     <ion-item text-wrap *ngFor="let announcement of announcements">
10       <h3>{{formatDatetime(announcement.localtime)}}</h3>
11       <p>{{announcement.message}}</p>
12     </ion-item>
13   </ion-list>
14 </ion-content>

```

Kode 3.5: *Content* pada Halaman *Announcement*

(b) Analisis Sistem Usulan pada Komponen *Announcements*

Pada komponen *announcement*, terdapat beberapa UI Component yang akan diimplementasikan seperti pada gambar 3.2b, diantaranya adalah sebagai berikut:

- *Content*

Komponen ini akan digunakan sebagai penyedia area konten yang digunakan untuk mengontrol area yang dapat digulir dan menampilkan isi konten dari halaman *announcement*. UI Component *Content* dengan tag `<ion-content>` pada Ionic Framework versi 6 tidak mengalami perubahan dari Ionic Framework versi 3.

- *Refresher*

Refresher menyediakan fungsionalitas pull-to-refresh pada komponen *content*. UI Component *Refresher* dengan tag `<ion-refresher>` dan `<ion-refresher-content>` pada Ionic Framework versi 6 tidak mengalami perubahan dari Ionic Framework versi 3.

- *List*

List dengan tag `<ion-list>` akan terdiri dari beberapa baris item `<ion-item>` yang berisi label `<ion-label>`. UI Component *List* dengan tag `<ion-list>`, `<ion-item>` dan `<ion-label>` pada Ionic Framework versi 6 tidak mengalami perubahan dari Ionic Framework versi 3.

- *Item*

Item dengan tag `<ion-item>` sejak Ionic 4 mengalami perubahan dibandingkan pada Ionic 3, yaitu wajib menambahkan *label* dengan tag `<ion-label>`. Pada aplikasi WSDC 2017 Bali saat ini yang menggunakan Ionic 3, tidak terdapat tag `<ion-label>` pada `<ion-item>` (Kode 3.6). Sedangkan sesuai dengan ketentuan yang berlaku, pada aplikasi yang akan dibangun yang menggunakan Ionic 6, akan menggunakan tag `<ion-label>` di dalam `<ion-item>` (Kode 3.7).

```

1 <ion-item text-wrap *ngFor="let announcement of announcements">
2   <h3>{{formatDatetime(announcement.localtime)}}</h3>
3   <p>{{announcement.message}}</p>
4 </ion-item>

```

Kode 3.6: Tag `<ion-item>` dengan Ionic 3 di Aplikasi WSDC 2017 Bali Saat Ini

```

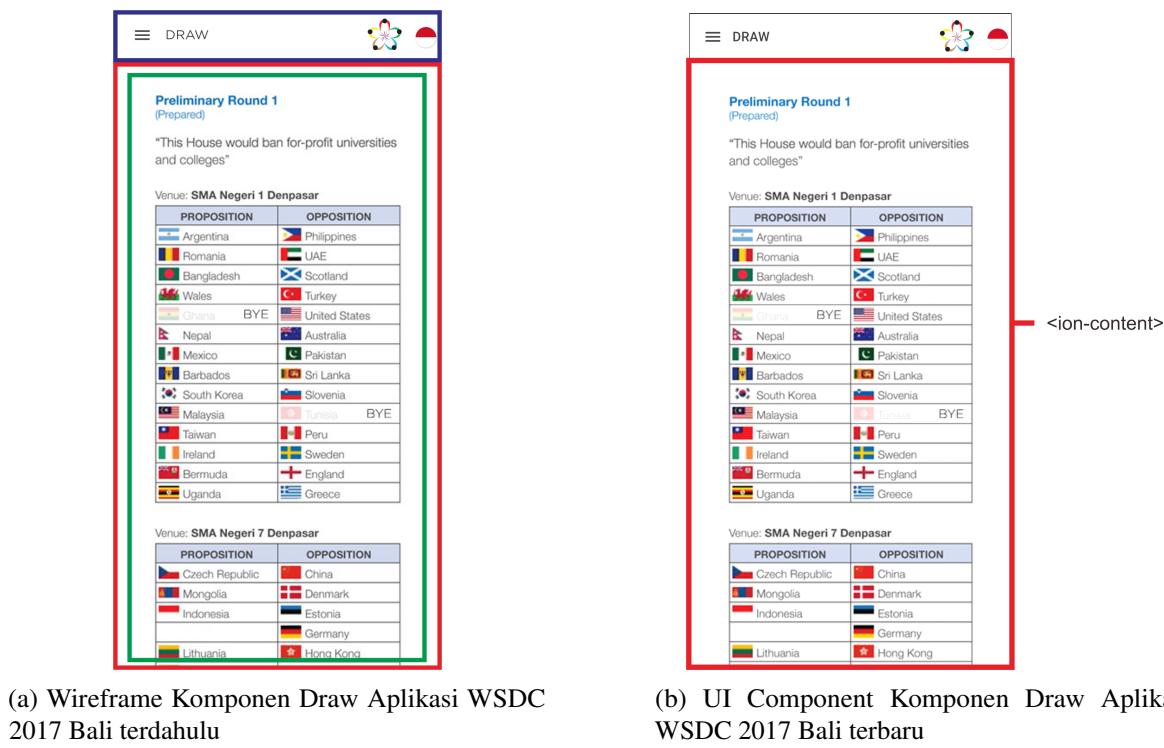
1 <ion-item color="wsdc-blue" *ngFor="let announcement of announcements;
2   let i = index">
3   <ion-label>
4     <h3>{{ formatDatetime(announcement.localtime) }}</h3>
5     <p>{{ announcement.message }}</p>
6   </ion-label>

```

Kode 3.7: Tag `<ion-item>` dengan Ionic 6 di Aplikasi WSDC 2017 Bali yang Akan dibuat

2. Komponen *Draw*

Komponen *Draw* digunakan untuk melihat pembagian kubu proposisi dan oposisi dari tiap-tiap negara peserta.



Gambar 3.3: Komponen Draw pada Aplikasi WSDC 2017 Bali

(a) Analisis Sistem Kini pada Komponen *Draw*

Terdapat *file* TypeScript, draw.ts, yang berfungsi untuk mengatur keseluruhan halaman. Di dalam *file* tersebut terdapat *decorator* @Component (Kode 3.8) dan *decorator* @ViewChild (Kode 3.9). Pada *decorator* @Component, terdapat CSS *selector* untuk memilih CSS mana yang akan digunakan, serta **templateUrl** untuk mendefinisikan ekxternal HTML *template* halaman *Draw* yang akan digunakan, yaitu draw.html. @ViewChild digunakan untuk memanggil elemen dari DOM untuk meamanggil komponen API ke dalam TypeScript, yaitu pada komponen *draw* adalah drawIFrame yang berada di *file* draw.html.

```

1  @Component ({
2    selector: 'page-draw',
3    templateUrl: 'draw.html'
4  })

```

Kode 3.8: @Component pada draw.ts

```

1  @ViewChild('drawIFrame') drawIFrame: ElementRef;

```

Kode 3.9: @ViewChild pada draw.ts

Terdapat kelas DrawPage yang berisi beberapa *method* yang akan digunakan di dalam aplikasi, diantaranya adalah sebagai berikut:

- ionViewDidLoad()

Method ini merupakan sebuah *lifecycle event* yang dijalankan saat halaman *draw* telah dimuat. *Event* ini hanya berjalan satu kali, jadi jika sudah melakukan chache terhadap halaman ini, maka ionViewDidLoad() tidak akan berjalan lagi. *Lifecycle event* ini berguna untuk melakukan pengaturan awal, yaitu untuk memuat data *draw* dari *storage*, kemudian data tersebut dimasukkan ke dalam *child* drawIFrame. Terakhir, method ini akan memanggil method presentLoading().

- presentLoading()

Method ini berfungsi untuk menampilkan sebuah *overlay* yang menunjukkan sebuah pesan

dan indikator pemuatan saat pertama kali halaman *draw* dimuat. Karena *overlay* ini muncul di atas konten aplikasi, maka aktivitas pengguna akan diblokir untuk sementara sampai seluruh halaman dimuat, yaitu sampai *method* `onDrawIframeLoad()` selesai.

- `onDrawIframeLoad()`

Method ini merupakan sebuah *template statement* dipanggil oleh *event* di dalam *tag* `<iFrame>` pada *draw.html* yaitu *event* (*load*). *Method* ini berfungsi untuk menampilkan data yang telah diambil yang disimpan di dalam *child* *drawIFrame*.

Terdapat *file* *draw.html* yang digunakan untuk menampilkan tata letak dari halaman *draw*. Terdapat beberapa komponen yang disediakan oleh Ionic Framework, yang diimplementasikan ke dalam halaman *draw*. Diantaranya adalah sebagai berikut:

- *Header*

Header dari halaman *draw* seperti pada gambar 3.3a menggunakan *tag* `<ion-header>` (Kode 3.10). *Tag* tersebut merupakan komponen *parent* yang menampung komponen *navbar* yang ditandai dengan kotak berwarna biru pada gambar 3.3a. Di dalam *navbar* tersebut, terdapat sebuah *tag* `<button>` untuk memunculkan *sidemenu*, `<ion-icon>` untuk menampilkan icon dari tombol pada *tag button*, dan *tag* `<ion-title>` sebagai judul dari halaman.

```

1 <ion-header>
2   <ion-navbar>
3     <button ion-button menuToggle>
4       <ion-icon name="menu"></ion-icon>
5     </button>
6     <ion-title>Draw</ion-title>
7   </ion-navbar>
8 </ion-header>

```

Kode 3.10: *Header* pada *draw.html*

- *Content*

Content dari halaman *draw* seperti pada gambar 3.3a menggunakan *tag* `<ion-content>` (Kode 3.11) yang ditandai menggunakan kotak berwarna merah. Di dalam *tag* ini terdapat sebuah *tag* `<iFrame>` yang berisi hasil pengundian grup untuk peserta WSDC 2017 Bali, ditandai menggunakan kotak berwarna hijau. *Tag* `<iFrame>` menampilkan hasil dari *method* `onDrawIframeLoad()` pada *draw.ts*.

```

1 <ion-content>
2   <iFrame #drawIFrame (load)="onDrawIframeLoad()" class="iframe-
  fullscreen"></iFrame>
3 </ion-content>

```

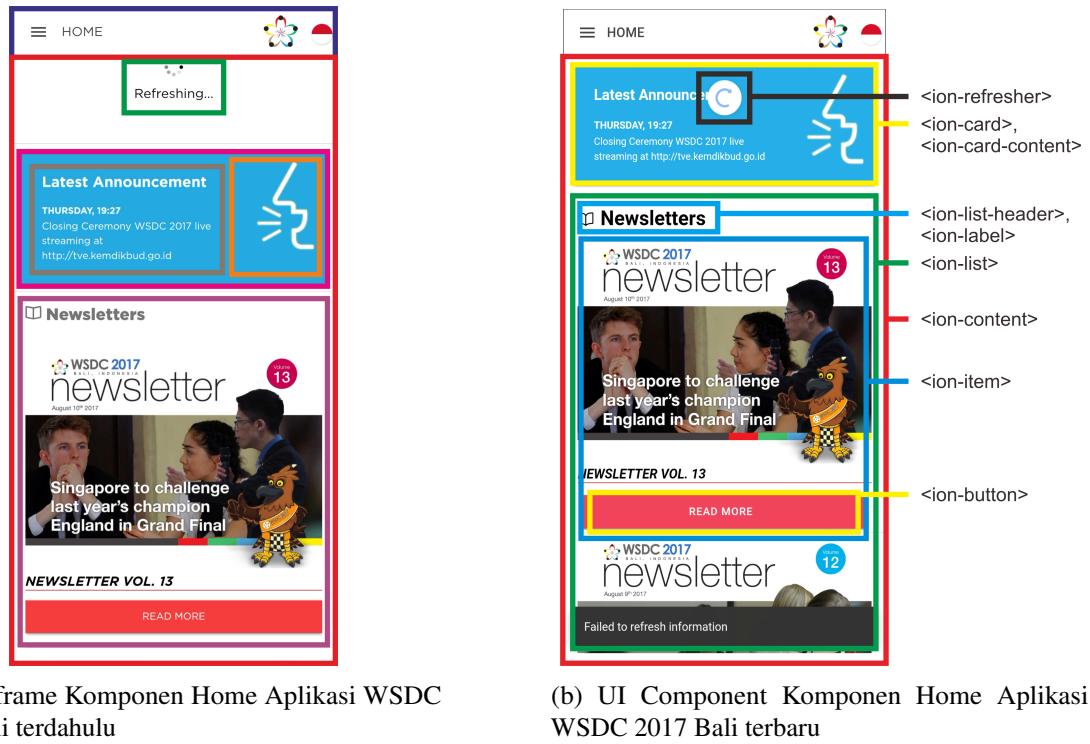
Kode 3.11: *Content* pada *draw.html*

(b) Analisis Sistem Usulan pada Komponen *Draw*

Pada komponen *draw*, terdapat sebuah UI Component, yaitu *Content* seperti pada gambar 3.3b. Komponen *content* akan digunakan sebagai penyedia area konten yang digunakan untuk mengontrol area yang dapat digulir dan menampilkan isi konten dari halaman *draw*. UI Component *Content* dengan *tag* `<ion-content>` pada Ionic Framework versi 6 tidak mengalami perubahan dari Ionic Framework versi 3.

3. Komponen *Home*

Komponen *Home* digunakan untuk menampilkan halaman utama aplikasi WSDC 2017 Bali yang berisi pengumuman terbaru dari acara WSDC 2017 Bali, dan berita-berita terkait acara WSDC 2017 Bali.



Gambar 3.4: Komponen Home pada Aplikasi WSDC 2017 Bali

(a) Analisis Sistem Kini pada Komponen *Home*

Komponen ini memiliki sebuah *file* TypeScript untuk mengatur keseluruhan halaman. Di dalam *file* home.ts terdapat sebuah *decorator* @Component untuk komponen (Kode 3.12). Di dalam decorator ini terdapat CSS *selector* untuk memilih CSS yang akan digunakan, serta templateUrl untuk mendefinisikan eksternal HTML *template* yang akan digunakan. *Template HTML* yang digunakan adalah *file* home.html.

```

1  @Component ({
2    selector: 'page-home',
3    templateUrl: 'home.html'
4  })

```

Kode 3.12: @Component pada home.ts

Komponen *Home* merupakan komponen yang menjadi rootPage dari aplikasi ini, yang dimasukan di dalam *file* app.component.ts. Maka dari itu, saat pertama kali aplikasi dijalankan, komponen *home*-lah yang pertama kali ditampilkan di dalam layar. rootPage di dalam *file* app.component.ts akan memanggil komponen *home*, yang kemudian *file* home.ts akan berjalan.

Di dalam *file* ini terdapat sebuah kelas HomePage yang berisi beberapa *method*, diantaranya adalah sebagai berikut:

- ionViewDidLoad()

Method ini merupakan sebuah *lifecycle event* yang dijalankan saat halaman *home* telah dimuat. *Event* ini hanya berjalan satu kali, jadi jika sudah melakukan cache terhadap halaman ini, maka ionViewDidLoad() tidak akan berjalan lagi. *Lifecycle event* ini berguna untuk melakukan pengaturan awal, yaitu untuk mengambil keseluruhan data aplikasi yang ada di dalam penyimpanan. Dengan memanfaatkan fitur *storage* yang dimiliki oleh Ionic Framework, *method* ini akan mengecek apakah sudah ada data dengan format .json yang berisi keseluruhan data aplikasi di dalam penyimpanan. Data tersebut berisi data *announcements*, *newsletters*, *schedule*, *venues*, *draws*, dan *info*. Jika data tersebut tidak ditemukan, maka akan diambil dari *file* wsdc-data.json dari aset lokal menggunakan HTTP API yang disediakan oleh Angular, yaitu HttpClient, kemudian dimasukan ke dalam

penyimpanan. Hal ini bertujuan jika pengguna tidak memiliki koneksi internet pada saat pemasangan aplikasi, aplikasi masih bisa dijalankan dan menampilkan halaman-halaman yang tidak kosong karena data diambil dari aset lokal.

Setelah itu, *method* ini mengambil data terbaru dari server dengan menggunakan Angular HttpClient. Jika sudah melewati batas waktu, dan aplikasi belum terhubung dengan server, maka *method* ini akan memanggil *method* showToast() yang akan menampilkan Toast yang berisi teks ‘Failed to refresh information’. Jika mengambil data dari server berhasil, maka data yang didapatkan dari server akan dimasukan ke dalam penyimpanan menggantikan data yang sudah ada di penyimpanan sebelumnya. Hal ini dilakukan dengan asumsi bahwa data yang terdapat di server merupakan data terbaru, sehingga data yang ada pada penyimpanan merupakan data lama dan harus diganti dengan data terbaru.

- **launch(url: string)**

Method ini merupakan sebuah *template statement* dipanggil oleh *event* di dalam tag <button> pada home.html yaitu *event* (click). *Method* ini memiliki sebuah parameter url yang bertipe *string*. Parameter tersebut berisi url dari berita yang akan dilihat oleh pengguna. Untuk membuka berita pada url tersebut memanfaatkan *plugin* InAppBrowser yang disediakan oleh Ionic.

- **formatDatetime(sqlDatetime: string)**

Method ini berfungsi untuk membuat format tanggal dan waktu. *Method* ini memiliki sebuah parameter, yaitu sqlDatetime yang bertipe *string*, yang merupakan sebuah *string* tanggal dengan format “tahun-bulan-hari jam-menit-detik”. *Method* ini akan mengembalikan sebuah teks yang berisi waktu, tanggal dan bulan.

- **doRefresh(refresher)**

Method ini berfungsi untuk melakukan penyegaran ulang pada halaman *home* untuk mendapatkan data *home* terbaru di dalam server, kemudian menyimpannya ke dalam penyimpanan. *Method* ini memiliki sebuah parameter refresher, yang berisi sebuah CustomEvent dari penyegaran ulang yang dilakukan. *Method* ini akan melakukan pemanggilan kembali kepada server, dalam batas waktu tertentu. Jika batas waktu maksimal telah tercapai, sedangkan server belum juga memberi tanggapan, maka akan memanggil *method* showToast() yang akan menampilkan sebuah Toast yang berisi teks ‘Failed to refresh information’. Jika berhasil untuk terhubung dengan server, *method* ini akan menghapus data yang berada di penyimpanan, dan digantikan dengan data yang telah didapatkan dari server.

- **showToast(message: string, duration: number = 3000)**

Method ini berfungsi untuk memunculkan sebuah native Toast, yaitu sebuah *popup* teks, dengan memanfaatkan UI Component milik Ionic Framework. *Method* ini menerima parameter berupa sebuah *string*, yang berisi pesan yang akan dimunculkan ke dalam sebuah Toast, dan memiliki sebuah parameter *duration* yang berisi lama waktu

- **onAnnouncementClick()**

Method ini merupakan sebuah *template statement* dipanggil oleh *event* di dalam tag <ion-card> pada home.html yaitu *event* (click). *Method* ini berfungsi untuk berpindah halaman menjadi halaman *announcement*.

File home.html digunakan untuk menampilkan tata letak dari halaman *home*. Terdapat beberapa komponen yang disediakan oleh Ionic Framework, yang diimplementasikan ke dalam halaman *home*. Diantaranya adalah sebagai berikut:

- **Header**

Halaman *home* memiliki *header* dengan tag <ion-header> (Kode 3.13) seperti pada gambar 3.4a yang ditandai dengan kotak berwarna biru. Tag tersebut merupakan komponen *parent* yang menampung komponen navbar yang ditandai dengan kotak berwarna biru. Di dalam navbar tersebut, terdapat sebuah tag <button> untuk memunculkan sidebar, <ion-icon> untuk menampilkan icon, dan tag <ion-title> sebagai judul dari halaman.

```

1 <ion-header>
2   <ion-navbar>
3     <button ion-button menuToggle>
4       <ion-icon name="menu"></ion-icon>
5     </button>
6     <ion-title>Home</ion-title>
7   </ion-navbar>
8 </ion-header>

```

Kode 3.13: *Header* pada home.html

- *Content*

Content pada halaman *home* dengan tag `<ion-content>` (Kode 3.14 pada gambar 3.4a) ditandai dengan kotak berwarna merah. Di dalam tag `<ion-content>` terdapat beberapa tag lainnya. Pertama yaitu sebuah tag `<ion-refresher>` yang digunakan untuk menampilkan simbol *refresh* saat pengguna menyegarkan halaman dengan cara melakukan *swipe* dari atas ke bawah layar, ditandai dengan kotak berwarna hijau. Terdapat tag `<ion-card>` yang digunakan sebagai tempat untuk pengumuman terkait acara WSDC 2017 Bali disimpan. Penggunaan *card* ditantai dengan kotak berwarna merah muda. Di dalam tag `<ion-card>` terdapat tag `<ion-grid>` untuk mengatur tata letak dari penyusunan isi dari suatu *card*. Di dalam *grid* tersebut terdapat satu baris dengan tag `<ion-row>` dan dua kolom dengan tag `<ion-col>`. Kolom pertama ditandai dengan kotak berwarna coklat, berisi tanggal beserta pengumuman, dan kolom kedua ditandai dengan kotak berwarna oranye berisi gambar. Selanjutnya terdapat sebuah tag `<ion-list>` untuk menyimpan berita-berita terkait acara WSDC 2017 Bali, yang ditandai dengan warna ungu. Di dalam *list* tersebut terdapat tag `<ion-list-header>` sebagai judul dari *list*, dan tag `<ion-item>` untuk menyimpan berita-berita terkait acara WSDC 2017 Bali. Di dalam tag `<ion-item>` terdapat tag `<button>` yang apabila ditekan oleh pengguna, maka akan mengarahkan pengguna untuk melihat berita tertentu sesuai dengan *item* yang dipilih dengan memanggil *method* `launch()` yang ada di *home.ts*.

```

1 <ion-content>
2   <ion-refresher (ionRefresh)="doRefresh($event)">
3     <ion-refresher-content pullingIcon="arrow-dropdown" pullingText="Pull
4       to refresh" refreshingSpinner="circles" refreshingText="Refreshing
5       ...">
6     </ion-refresher-content>
7   </ion-refresher>
8   <ion-card (click)="onAnnouncementClick()">
9     <ion-grid>
10    <ion-row>
11      <ion-col col-9>
12        <ion-card-header text-wrap>
13          Latest Announcement
14        </ion-card-header>
15        <ion-card-content>
16          <h3>{{formatDatetime(wsdcData?.announcements[0].localtime)
17 }}</h3>
18          <p>{{wsdcData?.announcements[0].message}}</p>
19        </ion-card-content>
20      </ion-col>
21      <ion-col col-3>
22        
23      </ion-col>
24    </ion-row>
25  </ion-grid>
26 </ion-card>
27 <ion-list>
28   <ion-list-header>

```

```

26     <ion-icon ios="ios-book-outline" md="md-book"></ion-icon>
27     Newsletters
28   </ion-list-header>
29   <ion-item *ngFor="let wsdcNews of wsdcData?.newsletters">
30     
32     <h2 text-wrap>{{wsdcNews.title}}</h2>
33     <button ion-button full block color="danger" (click)="launch(
34       wsdcNews.url)">Read More</button>
35   </ion-item>
36 </ion-list>
37 </ion-content>

```

Kode 3.14: *Content* pada home.html(b) Analisis Sistem Usulan pada Komponen *Home*

Pada komponen *Home*, terdapat beberapa UI Component yang akan diimplementasikan seperti pada gambar 3.4b, diantaranya adalah sebagai berikut:

- Content

Komponen ini akan digunakan sebagai penyedia area konten yang digunakan untuk mengontrol area yang dapat digulir dan menampilkan isi konten dari halaman *home*. UI Component *Content* dengan tag `<ion-content>` pada Ionic Framework versi 6 tidak mengalami perubahan dari Ionic Framework versi 3.

- *Refresher*

Refresher menyediakan fungsionalitas pull-to-refresh pada komponen *content*. UI Component *Refresher* dengan tag `<ion-refresher>` dan `<ion-refresher-content>` pada Ionic Framework versi 6 tidak mengalami perubahan dari Ionic Framework versi 3.

- *Card*

Komponen ini akan digunakan sebagai tampilan antar muka, yang dapat menjadi titik masuk ke dalam informasi yang lebih detail. UI Component *Card* dengan tag `<ion-card>`, `<ion-card-title>` dan `<ion-card-content>` pada Ionic Framework versi 6 tidak mengalami perubahan dari Ionic Framework versi 3.

- *Grid*

Komponen ini akan digunakan untuk membangun tata letak kustom pada halaman *home* bagian *announcement*, yang terdiri dari baris dan kolom. UI Component *Grid* dengan tag `<ion-grid>`, `<ion-row>` dan `<ion-col>` pada Ionic Framework versi 6 tidak mengalami perubahan dari Ionic Framework versi 3

- *List Header*

List Header dengan tag `<ion-list-header>` sejak Ionic 4 diwajibkan untuk selalu menambahkan tag `<ion-label>`. Pada aplikasi WSDC 2017 Bali saat ini yang menggunakan Ionic 3, tidak terdapat tag `<ion-label>` pada `<ion-list-header>` (Kode 3.15). Sedangkan sesuai dengan ketentuan yang berlaku, pada aplikasi yang akan dibangun yang menggunakan Ionic 6, akan menggunakan tag `<ion-label>` di dalam `<ion-item-header>` (Kode 3.16).

```

1 <ion-list-header>
2   <ion-icon ios="ios-book-outline" md="md-book"></ion-icon>
3   Newsletters
4 </ion-list-header>

```

Kode 3.15: Tag `<ion-list-header>` dengan Ionic 3 di Aplikasi WSDC 2017 Bali Saat Ini

```

1 <ion-list-header>
2   <ion-icon name="book-outline"></ion-icon>
3   <ion-label>Newsletters</ion-label>
4 </ion-list-header>

```

Kode 3.16: Tag `<ion-list-header>` dengan Ionic 6 di Aplikasi WSDC 2017 Bali yang Akan dibuat

- *Icon*

Komponen ini akan digunakan untuk menampilkan ikon pada halaman *home*. UI Component *List* dengan tag `<ion-icon>` pada Ionic Framework versi 6 tidak mengalami perubahan dari Ionic Framework versi 3.

- *Button*

Di dalam halaman *home*, komponen ini merupakan sebuah komponen yang dapat diklik untuk mengarahkan pengguna ke URL yang berisi berita terkait WSDC 2017 Bali. Pada aplikasi WSDC 2017 Bali saat ini yang menggunakan Ionic 3, komponen ini ditulis menggunakan tag `<button>` (Kode 3.17). Sejak Ionic Framework versi 4, terjadi perubahan dengan mengganti tag tersebut menjadi `<ion-button>` pada aplikasi yang akan dibangun yang menggunakan Ionic 6 (Kode 3.18).

```
1 <button ion-button full block color="danger" (click)="launch(wsdcNews.url)">Read More</button>
```

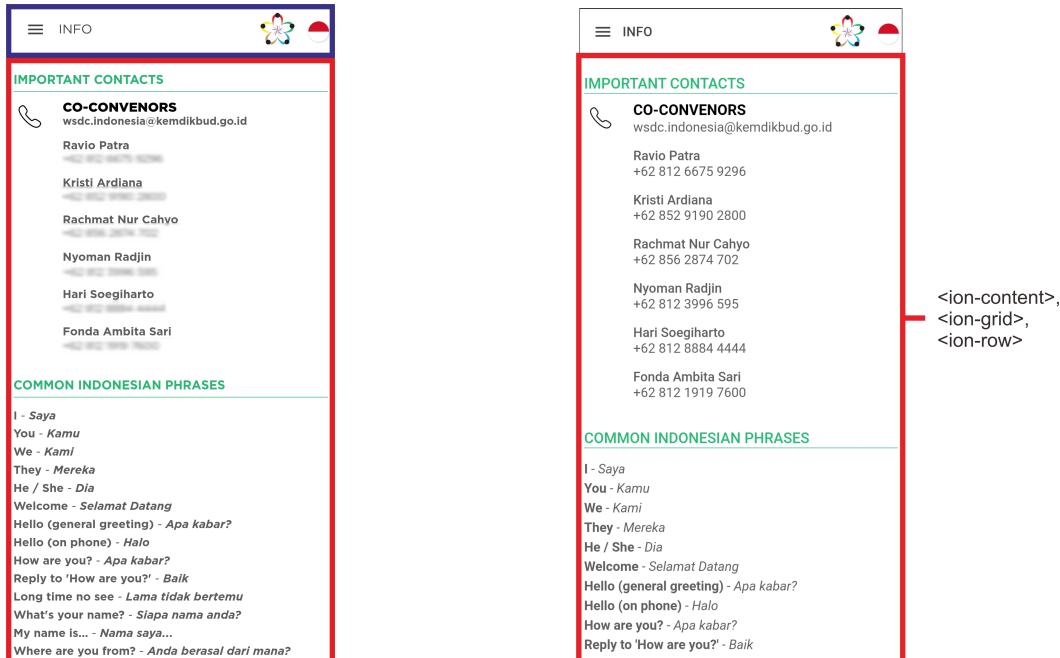
Kode 3.17: *Button* dengan Ionic 3 di Aplikasi WSDC 2017 Bali Saat Ini

```
1 <ion-button full block color="danger" (click)="launch(wsdcNews.url)">Read More</ion-button>
```

Kode 3.18: *Button* dengan Ionic 6 di Aplikasi WSDC 2017 Bali yang Akan dibuat

4. Komponen Info

Komponen Info digunakan untuk menampilkan informasi yang berisi kontak penting untuk acara WSDC 2017 Bali, kosa-kata dalam Bahasa Indonesia, serta *credits* kepada pengembang aplikasi.



(a) Wireframe Komponen Info Aplikasi WSDC 2017 Bali terdahulu

(b) UI Component Komponen Info Aplikasi WSDC 2017 Bali terbaru

Gambar 3.5: Komponen Info pada Aplikasi WSDC 2017 Bali

(a) Analisis Sistem Kini pada Komponen Info

Komponen ini memiliki sebuah file TypeScript untuk mengatur keseluruhan halaman. Di dalam file `info.ts` terdapat sebuah *decorator* `@Component` untuk komponen (Kode 3.19). Di dalam *decorator* ini terdapat CSS *selector* untuk memilih CSS yang akan digunakan, serta `templateUrl`

untuk mendefinisikan ekxternal HTML *template* yang akan digunakan. *Template HTML* yang digunakan adalah *file info.html*.

```

1 @Component({
2   selector: 'page-info',
3   templateUrl: 'info.html'
4 })

```

Kode 3.19: @Component pada info.ts

Terdapat kelas InfoPage pada komponen info. Kelas ini hanya berisi *constructor* yang digunakan untuk menginisialisai halaman yang akan digunakan. *Constructor* sendiri berfungsi untuk memuat data info dari penyimpanan dan memasukannya ke dalam sebuah variabel lokal. Terdapat *file info.html* yang digunakan untuk menampilkan tata letak dari halaman info. Terdapat beberapa komponen yang disediakan oleh Ionic Framework, yang diimplementasikan ke dalam halaman info. Diantaranya adalah sebagai berikut:

- *Header*

Halaman info memiliki *header* dengan tag `<ion-header>` (Kode 3.20) seperti pada gambar 3.5a. Tag tersebut merupakan komponen *parent* yang menampung komponen navbar yang ditandai dengan kotak berwarna biru. Di dalam navbar tersebut, terdapat sebuah tag `<button>` untuk memunculkan *sidemenu* dan tag `<ion-icon>` untuk menampilkan icon dari tombol pada tag *button*. Terdapat tag `<ion-title>` sebagai judul dari halaman, yaitu “Info”.

```

1 <ion-header>
2   <ion-navbar>
3     <button ion-button menuToggle>
4       <ion-icon name="menu"></ion-icon>
5     </button>
6     <ion-title>Info</ion-title>
7   </ion-navbar>
8 </ion-header>

```

Kode 3.20: *Header* pada info.html

- *Content*

Content pada halaman info memiliki tag `<ion-content>` (Kode 3.21) yang pada gambar 3.5a dengan kotak berwarna merah. Di dalam tag info terdapat tag `<ion-grid>` untuk mengatur *layout* dari *content*. Di dalam tag `<ion-grid>` terdapat sebuah tag `<ion-row>` yang berisi sebuah tag `<div>`. Tag tersebut berisi info yang di dapatkan pada *constructor* di *file info.ts*.

```

1 <ion-content>
2   <ion-grid>
3     <ion-row>
4       <div [innerHTML]=wsdcInfoData>
5         </div>
6     </ion-row>
7   </ion-grid>
8 </ion-content>

```

Kode 3.21: *Content* pada info.html

(b) Analisis Sistem Usulan pada Komponen Info

Pada komponen info, terdapat beberapa UI Component yang akan diimplementasikan seperti pada gambar 3.5b, diantaranya adalah sebagai berikut:

- *Content*

Komponen ini akan digunakan sebagai penyedia area konten yang digunakan untuk mengontrol area yang dapat digulir dan menampilkan isi konten dari halaman info. UI Component

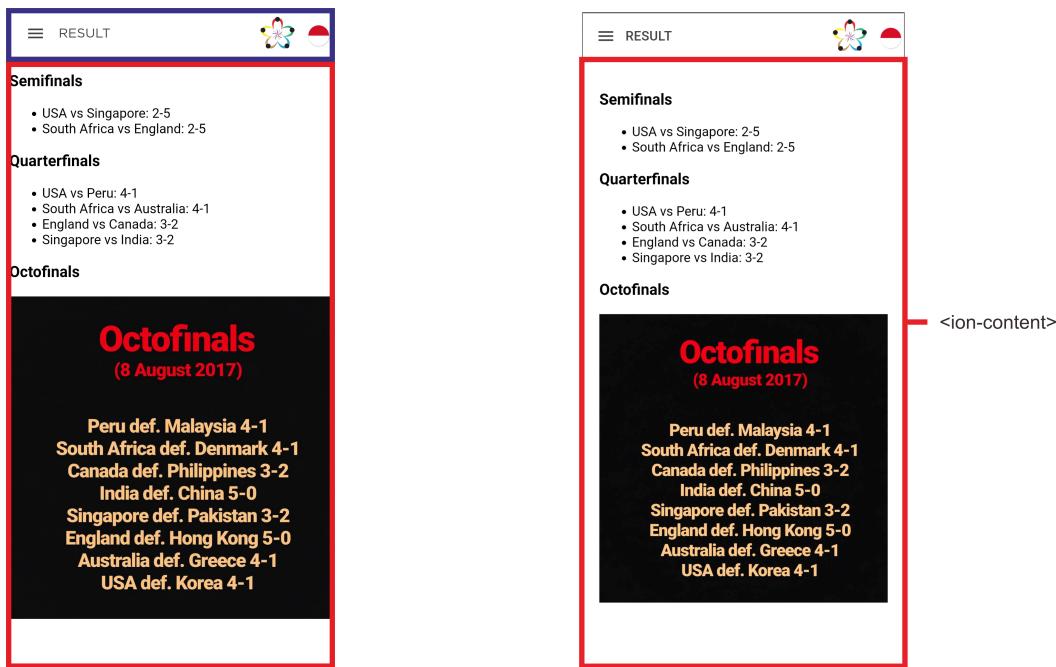
Content dengan tag `<ion-content>` pada Ionic Framework versi 6 tidak mengalami perubahan dari Ionic Framework versi 3.

- *Grid*

Komponen ini akan digunakan untuk membangun tata letak kustom pada halaman info, yang terdiri dari baris. UI Component *Grid* dengan tag `<ion-grid>` dan `<ion-row>` pada Ionic Framework versi 6 tidak mengalami perubahan dari Ionic Framework versi 3

5. Komponen *Result*

Komponen *Result* digunakan untuk melihat hasil dari pertandingan yang berisi pemenang dan skornya dari masing-masing babak pertandingan.



(a) Wireframe Komponen *Result* Aplikasi WSDC 2017 Bali terdahulu

(b) UI Component Komponen *Result* Aplikasi WSDC 2017 Bali terbaru

Gambar 3.6: Komponen *Result* pada Aplikasi WSDC 2017 Bali

(a) Analisis Sistem Kini pada Komponen *Result*

Komponen ini memiliki sebuah file TypeScript untuk mengatur keseluruhan halaman. Di dalam file `result.ts` terdapat sebuah *decorator* `@Component` untuk komponen (Kode 3.22) dan *decorator* `@ViewChild` (Kode 3.27). Di dalam *decorator* ini terdapat CSS *selector* untuk memilih CSS yang akan digunakan, serta `templateUrl` untuk mendefinisikan eksternal HTML *template* yang akan digunakan. *Template HTML* yang digunakan adalah file `info.html`. `@ViewChild` digunakan untuk memanggil elemen dari DOM untuk memanggil komponen API ke dalam TypeScript, yaitu pada komponen *result* adalah `resultIFrame` yang berada di file `result.html`.

```

1  @Component ({
2    selector: 'page-result',
3    templateUrl: 'result.html'
4  })

```

Kode 3.22: `@Component` pada `result.ts`

```

1  @ViewChild('resultIFrame') resultIFrame: ElementRef;

```

Kode 3.23: `@ViewChild` pada `result.ts`

Terdapat kelas ResultPage dengan beberapa *method* yang digunakan, yaitu:

- ionViewDidLoad()

Method ini merupakan sebuah *lifecycle event* yang dijalankan saat halaman *result* telah dimuat. *Event* ini hanya berjalan satu kali, jadi jika sudah melakukan cache terhadap halaman ini, maka ionViewDidLoad() tidak akan berjalan lagi. *Lifecycle event* ini berguna untuk melakukan pengaturan awal, yaitu untuk memuat data *result* yang sudah disimpan di dalam penyimpanan internal. Setelah berhasil memuat data *result*, data tersebut akan dimasukan ke dalam *child* resultIFrame. Terakhir, akan dipanggil *method* presentLoading().

- presentLoading()

Method ini berfungsi untuk menampilkan sebuah *overlay* yang menunjukkan sebuah pesan dan indikator pemuatan saat pertama kali halaman *draw* dimuat. Karena *overlay* ini muncul di atas konten aplikasi, maka aktivitas pengguna akan diblokir untuk sementara sampai seluruh halaman dimuat, yaitu sampai *method* onDrawIframeLoad() selesai.

- onResultIframeLoad()

Method ini merupakan sebuah *template statement* dipanggil oleh *event* di dalam tag <iFrame> pada result.html yaitu *event* (load). *Method* ini berfungsi untuk menampilkan data yang telah diambil yang disimpan di dalam *child* resultIFrame.

Terdapat file result.html yang digunakan untuk menampilkan tata letak dari halaman *result*. Terdapat beberapa komponen yang disediakan oleh Ionic Framework, yang diimplementasikan ke dalam halaman *result*. Diantaranya adalah sebagai berikut:

- *Header*

Halaman *result* memiliki *header* dengan tag <ion-header> (Kode 3.24) seperti pada gambar 3.6a. Tag tersebut merupakan komponen *parent* yang menampung komponen navbar yang ditandai dengan kotak berwarna biru. Di dalam navbar tersebut, terdapat sebuah tag <button> untuk memunculkan *sidemenu*, dan <ion-icon> untuk menampilkan icon dari tombol pada tag button. Terdapat tag <ion-title> sebagai judul dari halaman, yaitu “Result”.

```

1 <ion-header>
2   <ion-navbar>
3     <button ion-button menuToggle>
4       <ion-icon name="menu"></ion-icon>
5     </button>
6     <ion-title>Result</ion-title>
7   </ion-navbar>
8 </ion-header>
```

Kode 3.24: *Header* pada result.html

- *Content*

Content pada halaman *result* memiliki tag <ion-content> (Kode 3.25) yang pada gambar 3.6a dengan kotak berwarna merah. Di dalam tag <ion-content> terdapat tag <iFrame>. Tag tersebut berisi informasi mengenai daftar pemenang acara WSDC 2017 bali yang di dapatkan pada *method* onResultIframeLoad() di kelas ResultPage pada file result.ts.

```

1 <ion-content>
2   <iFrame #resultIFrame (load)="onResultIframeLoad()" class="iframe-
3     fullscreen"></iFrame>
4 </ion-content>
```

Kode 3.25: *Content* pada result.html

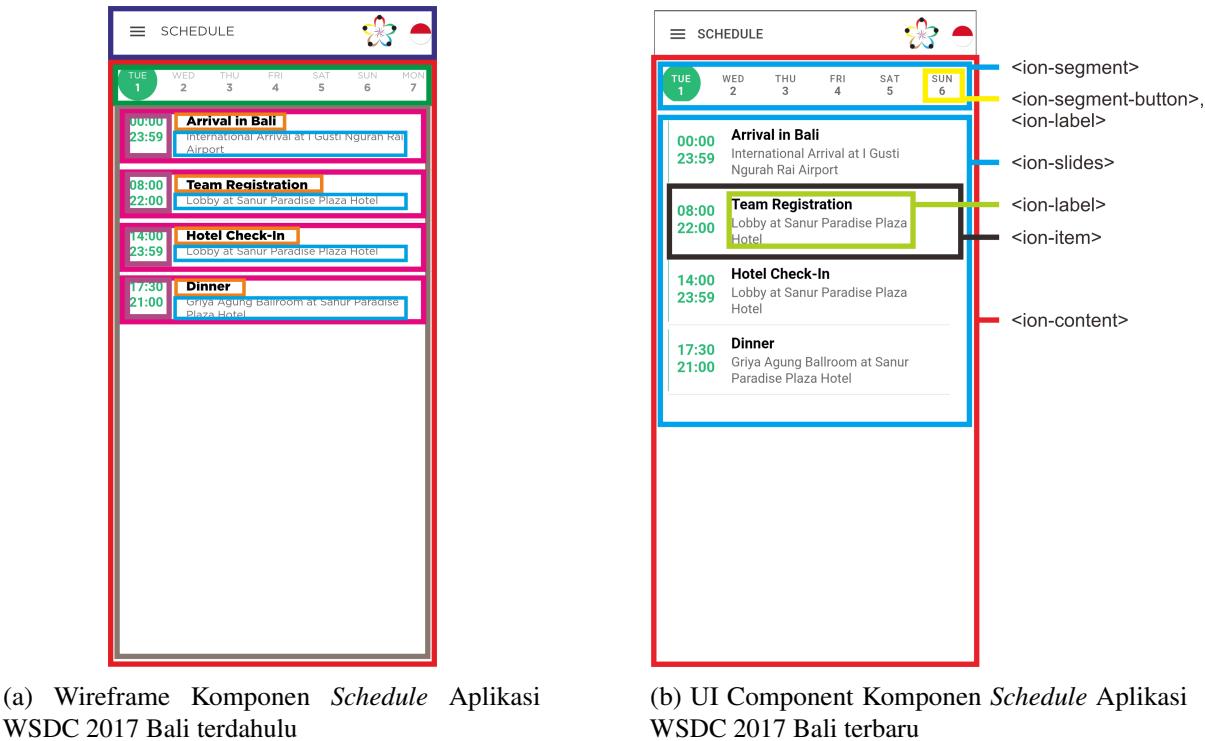
(b) Analisis Sistem Usulan pada Komponen *Result*

Pada komponen *Result*, terdapat sebuah UI Component, yaitu *Content* seperti pada gambar 3.6b. Komponen *content* akan digunakan sebagai penyedia area konten yang digunakan untuk mengontrol area konten dan menampilkan isi konten dari halaman *result*. UI Component *Content* dengan

tag <ion-content> pada Ionic Framework versi 6 tidak mengalami perubahan dari Ionic Framework versi 3.

6. Komponen *Schedule*

Komponen *Schedule* digunakan untuk melihat jadwal pertandingan yang berisi waktu, tanggal, hari, dan lokasi untuk setiap jadwal pertandingan WSDC 2017 Bali.



Gambar 3.7: Komponen *Schedule* pada Aplikasi WSDC 2017 Bali

(a) Analisis Sistem Kini pada Komponen *Schedule*

Komponen ini memiliki sebuah *file* TypeScript untuk mengatur keseluruhan halaman. Di dalam *file* schedule.ts terdapat sebuah *decorator* @Component untuk komponen (Kode 3.26) dan dua *decorator* @ViewChild (Kode 3.27). Di dalam decorator ini terdapat CSS *selector* untuk memilih CSS yang akan digunakan, serta **templateUrl** untuk mendefinisikan eksternal HTML *template* yang akan digunakan. *Template* HTML yang digunakan adalah *file* info.html. @ViewChild digunakan untuk memanggil elemen dari DOM untuk memanggil komponen API ke dalam TypeScript, yaitu pada komponen *result* adalah scheduleSlider dan segmentContainer yang berada di *file* result.html. scheduleSlider berfungsi untuk menyimpan konten dari sebuah *slide*. Sedangkan segmentContainer berfungsi untuk menyimpan konten dari sebuah *segment*.

```

1  @Component ({
2    selector: 'page-schedule',
3    templateUrl: 'schedule.html'
4  })

```

Kode 3.26: @Component pada schedule.ts

```

1  @ViewChild('scheduleSlider') slider: Slides;
2  @ViewChild('segmentContainer') segmentContainer: ElementRef;

```

Kode 3.27: @ViewChild pada schedule.ts

Terdapat kelas SchedulePage pada schedule.ts yang ini memiliki satu *constructor*. *Constructor* kelas ini berfungsi untuk mengambil data *result* yang berada di dalam penyimpanan internal.

Data tersebut kemudian disimpan ke dalam variabel lokal schedules. Kemudian akan mengatur *segment* yang aktif pada saat pertama kali halaman *result* dibuka, yaitu *segment* yang pertama dan menampilkan *slide* pertama yang berisi jadwal pada hari pertama. Terdapat beberapa *method* yang digunakan, yaitu:

- `onSegmentChanged(segmentButton)`

Merupakan sebuah *template statement* dipanggil oleh *event* di dalam tag `<ion-segment>` pada `schedule.html` yaitu *event* (`ionChange`). (`ionChange`) merupakan *event* yang dimiliki oleh UI Component `ion-segment` milik Ionic Framework. *Method* ini digunakan ketika pengguna memilih *segment* pada tag `<ion-segment>` di dalam file `schedule.html`. *Method* ini memiliki sebuah parameter `segmentButton` yang berisi *event* dari sebuah segment. *Child component* `slides` kemudian akan diubah sesuai dengan *value* yang ada pada parameter, yaitu hari yang sedang aktif.

- `onSlideChanged()`

Merupakan sebuah *template statement* dipanggil oleh *event* di dalam tag `<ion-slides>` pada `schedule.html` yaitu *event* (`ionSlideDidChange`). (`ionSlideDidChange`) merupakan *event* yang dimiliki oleh UI Component `ion-slides` milik Ionic Framework. *Method* ini berfungsi untuk berpindah antar `slides` saat pengguna menggeser `slides` tersebut ke kanan atau ke kiri layar.

- `getDayName(sqlDate: string)`

Method ini berfungsi untuk mengembalikan nama hari dari parameter.

- `getDate(sqlDate: string)`

Method ini bergunsi untuk mengembalikan tanggal dari parameter.

Terdapat file `schedule.html` yang digunakan untuk menampilkan halaman *schedule*. Terdapat beberapa komponen yang disediakan oleh Ionic Framework, yang diimplementasikan ke dalam halaman *schedule*. Diantaranya adalah sebagai berikut:

- *Header*

Halaman *schedule* memiliki *header* dengan tag `<ion-header>` (Kode 3.28) seperti pada gambar 3.7a. Tag tersebut merupakan komponen *parent* yang menampung komponen *navbar* yang ditandai dengan kotak berwarna biru. Di dalam *navbar* tersebut, terdapat sebuah tag `<button>` untuk memunculkan *sidemenu* dan `<ion-icon>` untuk menampilkan icon dari tombol pada tag *button*. Terdapat tag `<ion-title>` sebagai judul dari halaman, yaitu “Schedule”.

```

1 <ion-header>
2   <ion-navbar>
3     <button ion-button menuToggle>
4       <ion-icon name="menu"></ion-icon>
5     </button>
6     <ion-title>Schedule</ion-title>
7   </ion-navbar>
8 </ion-header>

```

Kode 3.28: *Header* pada `schedule.html`

- *Content*

Content pada halaman *result* memiliki tag `<ion-content>` (Kode 3.29) yang pada gambar 3.7a dengan kotak berwarna merah. Di dalam tag `<ion-content>` terdapat dua buah tag `<div>` yang masing masing berisi tag `<ion-segment>` dan tag `<ion-slides>`. Tag `<ion-segment>` digunakan untuk tampilan hari, seperti pada gambar 3.7a yang ditandai dengan kotak berwarna hijau. Tag `<ion-slides>` digunakan untuk tampilan jadwal di dalam satu hari, seperti yang ditandai dengan kotak berwarna coklat.

Setiap jadwal yang berada di tag `<ion-slides>` dibungkus dengan tag `<ion-list>` seperti pada kotak berwarna muda di gambar 3.7a. Dalam satu tag `<ion-list>` terdapat tag `<ion-note>` yang berisi waktu mulai dan waktu selesai dari satu jadwal seperti yang ditandai dengan kotak berwarna ungu, tag `<h3>` berisi nama acara seperti

yang ditandai dengan kotak berwarna oranye, dan tag <p> yang berisi tempat acara tersebut diadakan ditandai dengan kotak berwarna biru muda.

```

1 <ion-content>
2   <div id="schedulesContainer">
3     <div id="schedulesSegments">
4       <ion-segment #segmentContainer *ngIf="schedules" [(ngModel)]="selectedSegmentIdx" (ionChange)="onSegmentChanged($event)">
5         <ion-segment-button *ngFor="let schedule of schedules; let i = index" [value]="i">
6           <div class="day">{{getDayName(schedule.date)}}</div>
7           <div class="date">{{getDate(schedule.date)}}</div>
8         </ion-segment-button>
9       </ion-segment>
10      </div>
11      <div id="schedulesSlides">
12        <ion-slides #scheduleSlider (ionSlideDidChange)="onSlideChanged()">
13          <ion-slide *ngFor="let schedule of schedules">
14            <ion-list>
15              <ion-item text-wrap *ngFor="let agenda of schedule.agenda">
16                <ion-note item-start>
17                  {{agenda.start}}<br/>
18                  {{agenda.end}}
19                </ion-note>
20                <h3>{{agenda.title}}</h3>
21                <p>{{agenda.subtitle}}</p>
22              </ion-item>
23            </ion-list>
24          </ion-slide>
25        </ion-slides>
26      </div>
27    </div>
28  </ion-content>
```

Kode 3.29: *Content* pada schedule.html

(b) Analisis Sistem Usulan pada Komponen *Schedule*

Pada komponen *schedule*, terdapat beberapa UI Component yang akan diimplementasikan seperti pada gambar 3.7b, diantaranya adalah sebagai berikut:

- *Content*
Komponen ini akan digunakan sebagai penyedia area konten yang digunakan untuk mengontrol area yang dapat digulir dan menampilkan isi konten dari halaman *schedule*. UI Component *Content* dengan tag <ion-content> pada Ionic Framework versi 6 tidak mengalami perubahan dari Ionic Framework versi 3.

- *Item*
Item dengan tag <ion-item> sejak Ionic 4 mengalami perubahan dibandingkan pada Ionic 3, yaitu wajib menambahkan *label* dengan tag <ion-label>. Pada aplikasi WSDC 2017 Bali saat ini yang menggunakan Ionic 3, tidak terdapat tag <ion-label> pada <ion-item> (Kode 3.30). Sedangkan sesuai dengan ketentuan yang berlaku, pada aplikasi yang akan dibangun yang menggunakan Ionic 6, akan menggunakan tag <ion-label> di dalam <ion-item> (Kode 3.31).

```

1 <ion-item text-wrap *ngFor="let agenda of schedule.agenda">
2   <ion-note item-start>
3     {{agenda.start}}<br/>
4     {{agenda.end}}
5   </ion-note>
6   <h3>{{agenda.title}}</h3>
7   <p>{{agenda.subtitle}}</p>
8 </ion-item>
```

Kode 3.30: Tag `<ion-item>` dengan Ionic 3 di Aplikasi WSDC 2017 Bali Saat Ini

```

1 <ion-item *ngFor="let agenda of schedule.agenda;" >
2   <ion-note item-start>
3     {{agenda.start}}<br/>
4     {{agenda.end}}
5   </ion-note>
6   <ion-label class="ion-text-wrap">
7     <h3>{{agenda.title}}</h3>
8     <p>{{agenda.subtitle}}</p>
9   </ion-label>
10 </ion-item>

```

Kode 3.31: Tag `<ion-item>` dengan Ionic 6 di Aplikasi WSDC 2017 Bali yang Akan dibuat

- *List*

List berfungsi untuk menyimpan konten yang terdiri dari beberapa baris. *List* dengan tag `<ion-list>` akan terdiri dari beberapa baris item `<ion-item>` dan akan memiliki sebuah *header*. UI Component *List* dengan tag `<ion-list>`, dan `<ion-item>` pada Ionic Framework versi 6 tidak mengalami perubahan dari Ionic Framework versi 3.

- *Segment*

Komponen ini akan digunakan untuk pengguna agar dapat berpindah tampilan di dalam halaman yang sama. Seperti pada tampilan halaman jadwal yang ada pada aplikasi WSDC 2017 Bali saat ini, dimana pengguna dapat berpindah hari untuk mengetahui jadwal kegiatan pada hari tertentu yang dipilih oleh pengguna, namun masih berada di halaman yang sama, yaitu halaman Schedule. UI Component *Segment* dengan tag `<ion-segment>` pada Ionic Framework versi 6 tidak mengalami perubahan dari Ionic Framework versi 3. Sedangkan tag `<ion-segment-button>` yang berada di dalam tag `<ion-segment>` sejak Ionic 4 mengalami perubahan yaitu wajib menambahkan *label* dengan tag `<ion-label>`. Pada aplikasi WSDC 2017 Bali saat ini yang menggunakan Ionic 3, tidak terdapat tag `<ion-label>` pada `<ion-item>` (Kode 3.32). Sedangkan sesuai dengan ketentuan yang berlaku, pada aplikasi yang akan dibangun yang menggunakan Ionic 6, akan menggunakan tag `<ion-label>` di dalam `<ion-item>` (Kode 3.33).

```

1 <ion-segment-button *ngFor="let schedule of schedules; let i = index" [
  value]="i">
2   <div class="day">{{getDayName(schedule.date)}}</div>
3   <div class="date">{{getDate(schedule.date)}}</div>
4 </ion-segment-button>

```

Kode 3.32: Tag `<ion-segment-button>` dengan Ionic 3 di Aplikasi WSDC 2017 Bali Saat Ini

```

1 <ion-segment-button *ngFor="let schedule of schedules; let i = index" [
  value]="i">
2   <ion-label>
3     <div class="day">{{getDayName(schedule.date)}}</div>
4     <div class="date">{{getDate(schedule.date)}}</div>
5   </ion-label>
6 </ion-segment-button>

```

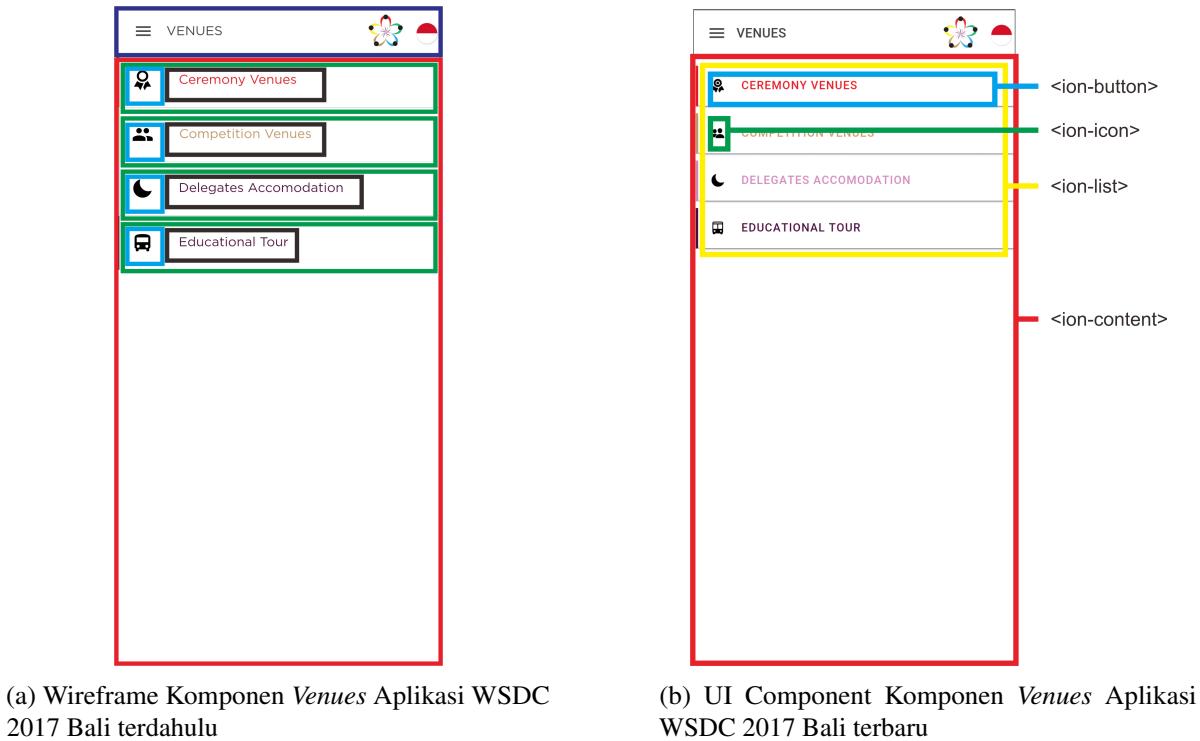
Kode 3.33: Tag `<ion-segment-button>` dengan Ionic 6 di Aplikasi WSDC 2017 Bali yang Akan Dibuat

- *Slides*

Komponen ini akan digunakan sebagai wadah dari *multi-section*. Penggunaan slide di halaman *schedule* yaitu untuk berpindah jadwal perhari dengan cara melakukan *swipe* dari kanan ke kiri layar atau sebaliknya. UI Component *Slides* dengan tag `<ion-slides>` pada Ionic Framework versi 6 tidak mengalami perubahan dari Ionic Framework versi 3.

7. Komponen *Venues*

Komponen *Venues* digunakan untuk menampilkan halaman *venues* yang berisi kategori dari masing-masing *venues*. Tiap-tiap kategori dapat ditekan untuk menampilkan halaman *venues maps*.



Gambar 3.8: Komponen *Venues* pada Aplikasi WSDC 2017 Bali

(a) Analisis Sistem Kini pada Komponen *Venues*

Komponen ini memiliki sebuah *file* TypeScript untuk mengatur keseluruhan halaman. Di dalam *file* *venues.ts* terdapat sebuah *decorator* `@Component` untuk komponen (Kode 3.34). Di dalam *decorator* ini terdapat CSS *selector* untuk memilih CSS yang akan digunakan, serta **templateUrl** untuk mendefinisikan eksternal HTML *template* yang akan digunakan. *Template HTML* yang digunakan adalah *file* *venues.html*

```

1 @Component ({
2   selector: 'page-venues',
3   templateUrl: 'venues.html'
4 })

```

Kode 3.34: `@Component` pada *venues.ts*

Terdapat kelas *VenuesPage* yang memiliki satu *constructor*. *Constructor* kelas ini berfungsi untuk mengambil data *venues* yang berada di dalam penyimpanan internal. Data tersebut kemudian disimpan ke dalam variabel lokal *valVenues*. Terdapat sebuah *method* *itemTapped()* yang berfungsi untuk berpindah halaman ke halaman *venues-map*, yang akan menampilkan peta lokasi berlangsungnya acara, sesuai dengan *venues* yang dipilih.

Terdapat *file* *venues.html* yang digunakan untuk menampilkan halaman *venues*. Terdapat beberapa komponen yang disediakan oleh Ionic Framework, yang diimplementasikan ke dalam halaman *venues*. Diantaranya adalah sebagai berikut:

- *Header*

Halaman *venues* memiliki *header* dengan tag `<ion-header>` (Kode 3.35) seperti pada gambar 3.8a. Tag tersebut merupakan komponen *parent* yang menampung komponen navbar yang ditandai dengan kotak berwarna biru. Di dalam navbar tersebut, terdapat sebuah tag

<button> untuk memunculkan *sidemenu*. Terdapat tag <ion-title> sebagai judul dari halaman, yaitu “Venues”.

```

1 <ion-header>
2   <ion-navbar>
3     <button ion-button menuToggle>
4       <ion-icon name="menu"></ion-icon>
5     </button>
6     <ion-title>Venues</ion-title>
7   </ion-navbar>
8 </ion-header>
```

Kode 3.35: *Header* pada venues.html

- *Content*

Content pada halaman venues memiliki tag <ion-content> (Kode 3.36) yang pada gambar 3.8a dengan kotak berwarna merah. Di dalam tag <ion-content> terdapat sebuah tag <ion-grid> dan sebuah tag <ion-row>. Di dalam tag <ion-row> terdapat sebuah tag <ion-list> yang berisi tag <ion-button> yang ditandai dengan kotak berwarna hijau pada gambar 3.8a. Masing-masing tag <ion-button> berisi tag <ion-icon> yang ditandai dengan kotak berwarna biru muda, dan tag berisi nama *venues* yang ditandai dengan kotak berwarna hitam.

```

1 <ion-content>
2   <ion-grid>
3     <ion-row>
4       <ion-list style="width: 100%;" no-lines>
5         <button ion-item id="{{wsdcVenue.id}}" *ngFor="let wsdcVenue
6           of venuesData" (click)="itemTapped($event, wsdcVenue)">
7           <ion-icon ios="ios-{{wsdcVenue.icon}}-outline" md="md-{{
8             wsdcVenue.icon}}" item-start></ion-icon>
9           <span>{{wsdcVenue.name}}</span>
10          </button>
11        </ion-list>
12      </ion-row>
13    </ion-grid>
14  </ion-content>
```

Kode 3.36: *Content* pada venues.html

Terdapat kelas VenuesPage pada venues.ts. Kelas ini memiliki satu *constructor*. *Constructor* kelas ini berfungsi untuk mengambil data *venues* yang berada di dalam penyimpanan. Data tersebut kemudian disimpan ke dalam variabel lokal *venuesData*, yang berisi *id*, *name*, *icon*, *geojson*, dan *colorIdx*. Terdapat sebuah *method* yaitu *itemTapped(event, wsdcVenue)*. *Method* ini memiliki dua buah parameter, *event* yang berisi *event* pada tag *button*, dan *wsdcVenue* yang merupakan data bertipe json yang berisi data lengkap sebuah venue yang ada di penyimpanan sesuai dengan data venue pada *event* di dalam tag *button*. Kemudian, dengan menggunakan NavController milik Ionic Framework, data *wsdcVenue* dikirimkan ke halaman Venues Map. Setelah itu halaman akan berpindah ke halaman Venues Map.

(b) Analisis Sistem Usulan pada Komponen *Venues*

Pada komponen *venues*, terdapat beberapa UI Component yang akan diimplementasikan seperti pada gambar 3.8b, diantaranya adalah sebagai berikut:

- *Content*

Komponen Content dengan tag <ion-content> akan digunakan sebagai penyedia area konten yang digunakan untuk mengontrol area yang dapat digulir dan menampilkan isi konten dari halaman *venues*. UI Component *Content* pada Ionic Framework versi 6 tidak mengalami perubahan dari Ionic Framework versi 3.

- *Grid*

Komponen ini akan digunakan untuk membangun tata letak kustom pada halaman info. UI

Component *Grid* dengan tag `<ion-grid>` dan `<ion-row>` pada Ionic Framework versi 6 tidak mengalami perubahan dari Ionic Framework versi 3

- *List*

List dengan tag `<ion-list>`, yang terdiri dari baris yang setiap barisnya berisi kategori *venues* yang disusun menggunakan *button*. UI Component *List* dengan tag `<ion-list>` pada Ionic Framework versi 6 tidak mengalami perubahan dari Ionic Framework versi 3.

- *Button*

Button digunakan untuk berpindah ke halaman *Venues Map* sesuai dengan tombol apa yang dipilih. Pada aplikasi WSDC 2017 Bali saat ini yang menggunakan Ionic 3, komponen ini ditulis menggunakan tag `<button>` (Kode 3.37). Sejak Ionic Framework versi 4, terjadi perubahan dengan mengganti tag tersebut menjadi `<ion-button>` pada aplikasi yang akan dibangun yang menggunakan Ionic 6 (Kode 3.38).

```

1 <button ion-item id="{{wsdcVenue.id}}" *ngFor="let wsdcVenue of
  venuesData" (click)="itemTapped($event, wsdcVenue)">
2   <ion-icon ios="ios-{{wsdcVenue.icon}}-outline" md="md-{{wsdcVenue.icon
    }}" item-start></ion-icon>
3   <span>{{wsdcVenue.name}}</span>
4 </button>

```

Kode 3.37: *Button* dengan Ionic 3 di Aplikasi WSDC 2017 Bali Saat Ini

```

1 <ion-button ion-item id="{{wsdcVenue.id}}" *ngFor="let wsdcVenue of
  venuesData" (click)="itemTapped($event, wsdcVenue)">
2   <ion-icon ios="ios-{{wsdcVenue.icon}}-outline" md="md-{{wsdcVenue.icon
    }}" item-start></ion-icon>
3   <span>{{wsdcVenue.name}}</span>
4 </ion-button>

```

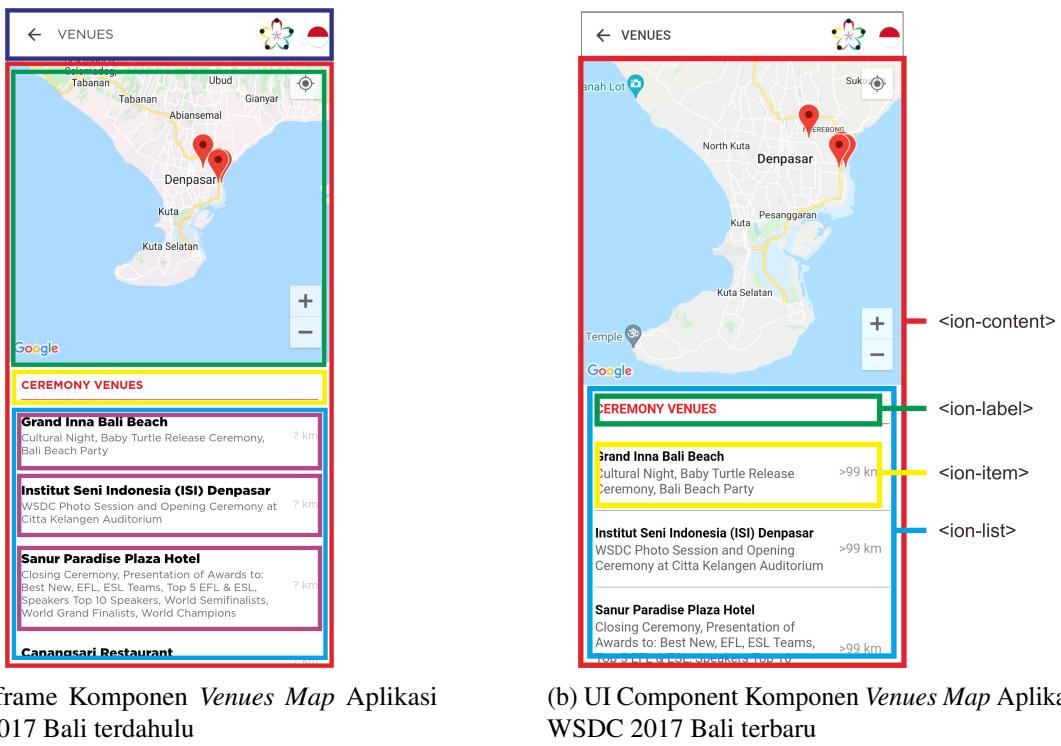
Kode 3.38: *Button* dengan Ionic 6 di Aplikasi WSDC 2017 Bali yang Akan Dibuat

- *Icon*

Komponen ini akan digunakan untuk menampilkan ikon pada halaman *venues*. UI Component *Icon* dengan tag `<ion-icon>` pada Ionic Framework versi 6 tidak mengalami perubahan dari Ionic Framework versi 3.

8. Komponen *Venues Map*

Komponen *Venues Map* digunakan untuk menampilkan halaman *Venues Map* yang berisi peta untuk masing-masing lokasi *venue*, serta jarak dari posisi pengguna ke tiap-tiap lokasi *venue*.

Gambar 3.9: Komponen *Venues Map* pada Aplikasi WSDC 2017 Bali(a) Analisis Sistem Kini pada Komponen *Venues Map*

Komponen *Venues Map* menampilkan sebuah peta yang berisi lokasi dari acara WSDC 2017 Bali. Komponen ini memiliki sebuah file TypeScript untuk mengatur keseluruhan halaman. Di dalam file *venues_map.ts* terdapat sebuah *decorator* @Component untuk komponen (Kode 3.39). Di dalam *decorator* ini terdapat CSS *selector* untuk memilih CSS yang akan digunakan, serta *templateUrl* untuk mendefinisikan eksternal HTML *template* yang akan digunakan. *Template* HTML yang digunakan adalah file *venues_map.html*

```

1  @Component({
2    selector: 'page-venuesmap',
3    templateUrl: 'venues_map.html',
4  })

```

Kode 3.39: @Component pada *venues_map.ts*

Terdapat kelas *VenuesMapPage* di dalam *app.module.ts*. Kelas ini memiliki sebuah *constructor*. *Constructor* kelas ini berfungsi untuk mengambil data *venues* yang berada di dalam penyimpanan internal, kemudian disimpan ke dalam variabel lokal *venuesData*. Di dalam *constructor* mengambil data yang dikirimkan oleh halaman *Venues*. Data tersebut kemudian dimasukkan ke dalam variabel lokal bernama *items*.

Pada komponen ini, terdapat sebuah *plugin* Google Maps, yang digunakan untuk menampilkan peta yang berisi lokasi dari kegiatan WSDC 2017 Bali. *Plugin* yang digunakan adalah *plugin* Google Maps yang disediakan oleh Cordova. *Plugin* tersebut diinisialisasikan di dalam *constructor*. Terdapat juga sebuah *plugin* Geolocation, yang berfungsi untuk menerima masukan posisi dari lokasi pengguna yang berisi *latitude* dan *longitude*, yang kemudian keseluruhan lokasi tersebut pada *constructor* akan dihitung jaraknya dari lokasi pengguna saat ini.

Terdapat beberapa *method* yang digunakan, diantaranya yaitu:

- *ngAfterViewInit()*

Method ini dipanggil hanya sekali ketika Angular menyelesaikan inisialisasi tampilan komponen. *Method* ini digunakan untuk menambahkan atribut ke dalam judul dari halaman, yaitu menambahkan warna pada teks judul.

- `loadMap()`

Method ini dipanggil di dalam *constructor*, dan berfungsi untuk menampilkan peta dengan bantuan *plugin* Google Maps. Pada *method* ini, peta pertama kali akan dibuat dengan pengaturan kamera yang mengarah ke lokasi latitude dan longitude dari kota Kuta, Bali. *Plugin* Google Maps sendiri akan memanfaatkan fitur-fitur *native* dari suatu perangkat. Fitur-fitur tersebut adalah untuk melakukan *gesture* seperti *scroll*, *tilt*, *rotate*, dan *zoom*. Terdapat fitur untuk menagkses kontrol pada Google Maps, seperti mengakses kompas, tombol lokasi pengguna saat ini, dan melihat peta di dalam ruangan. Saat Google Maps sudah tersedia untuk digunakan, *method* ini akan memanggil *method* `loadMarkers()` untuk membuat penanda pada peta.

- `loadMarkers()`

Method ini dipanggil oleh *method* `loadMap()`. *Method* ini digunakan untuk menampilkan *marker* dari setiap lokasi acara WSDC 2017 Bali yang sudah tersimpan di dalam variabel `items`. Google Maps menggunakan lokasi latitude dan longitude dari suatu lokasi yang berada di variabel `items` untuk membuat *marker*.

- `featTapped(event, index)`

Merupakan sebuah *template statement* dipanggil oleh *event* di dalam *tag* `<ion-item>` pada `venues_map.html` yaitu *event* (*click*). Saat pengguna melakukan klik di dalam *tag* `<ion-item>`, maka peta akan melakukan *zoom* sesuai dengan lokasi yang ada pada *tag* `<ion-item>`.

- `computeDistance(p1, p2)`

Method ini digunakan untuk menghitung jarak antara pengguna ke lokasi *venues*. *Method* ini memanfaatkan fitur dari *plugin* Google Maps, yaitu `computeDistanceBetween` dengan parameter lokasi *venues* dan lokasi perangkat pengguna yang didapatkan dari paramter *method* ini.

Terdapat *file* `venues_map.html` yang digunakan untuk menampilkan halaman *venues map*. Terdapat beberapa komponen yang disediakan oleh Ionic Framework, yang diimplementasikan ke dalam halaman. Diantaranya adalah sebagai berikut:

- *Header*

Halaman *venues* memiliki *header* dengan *tag* `<ion-header>` (Kode 3.40) seperti pada gambar 3.9a. *Tag* tersebut merupakan komponen *parent* yang menampung komponen navbar seperti yang ditandai dengan kotak berwarna biru. Di dalam navbar, terdapat sebuah *tag* `<button>` untuk memunculkan *sitemenu* dan `<ion-icon>` untuk menampilkan icon. Terdapat *tag* `<ion-title>` sebagai judul dari halaman, yaitu “Venues”.

```

1 <ion-header>
2   <ion-navbar>
3     <button ion-button menuToggle>
4       <ion-icon name="menu"></ion-icon>
5     </button>
6     <ion-title>Venues</ion-title>
7   </ion-navbar>
8 </ion-header>

```

Kode 3.40: *Header* pada `venues_map.html`

- *Content*

Content pada halaman *venues* dibungkus oleh *tag* `<ion-content>` (Kode 3.41) yang pada gambar 3.8a dengan kotak berwarna merah. Di dalam *tag* `<ion-content>` terdapat sebuah *tag* `<div>` dengan id bernilai `map`, untuk menampilkan peta lokasi dari *venues* seperti yang ditandai dengan kotak berwarna hijau pada gambar 3.9a. Untuk judul dari *venues* ditandai dengan kotak berwarna kuning dengan menggunakan *tag* `<h3>`. Terdapat sebuah *tag* `<ion-scroll>` seperti yang ditandai dengan kotak berwarna biru muda, berfungsi untuk menampilkan sebuah konten yang dapat digulir. Di dalam *tag* `<ion-scroll>` terdapat sebuah *tag* `<ion-list>` dan `<ion-item>` seperti yang ditandai dengan kotak

berwarna ungu, berisi nama, deskripsi, serta jarak pengguna ke tempat *venues* berada. Tag `<ion-item>` akan melakukan perulangan dengan menggunakan `*ngFor` yang tersedia pada Angular untuk menampilkan daftar *venues*.

```

1 <ion-content>
2   <div #map id="map"></div>
3   <h3 #pagetitle>
4     {{selectedItem.name}}
5   </h3>
6   <ion-scroll scrollY="true">
7     <ion-list>
8       <ion-item text-wrap *ngFor="let feature of items; let i=index" (
9         click)="featTapped($event, i)">
10        <h2>{{feature.name}}</h2>
11        <p>{{feature.description}}</p>
12        <ion-note item-end>
13          {{feature.distance}}
14        </ion-note>
15      </ion-item>
16    </ion-list>
17  </ion-scroll>
18 </ion-content>

```

Kode 3.41: Content pada venues_map.html

(b) Analisis Sistem Kini pada Komponen *Venues Map*

Pada komponen *venues map*, terdapat beberapa UI Component yang akan diimplementasikan seperti pada gambar 3.9b, diantaranya adalah sebagai berikut:

- Content

Komponen ini akan digunakan sebagai penyedia area konten yang digunakan untuk mengontrol area yang dapat digulir dan menampilkan isi konten dari halaman *venues_map*. UI Component *Content* dengan tag `<ion-content>` pada Ionic Framework versi 6 tidak mengalami perubahan dari Ionic Framework versi 3.

- Scroll

Tag `<ion-scroll>` telah dihapus sejak Ionic Framework versi 4, dan digantikan penggunaannya dengan hanya cukup menggunakan tag `<ion-content>` sejak Ionic Framework versi 4.

- List

List dengan tag `<ion-list>`, yang terdiri dari baris yang setiap barisnya berisi nama dan lokasi *venues* yang disusun menggunakan `<ion-item>`. UI Component *List* dengan tag `<ion-list>` dan `<ion-item>` pada Ionic Framework versi 6 tidak mengalami perubahan dari Ionic Framework versi 3.

Sebagai penyedia *interface* untuk mengakses SDK *native* dan API *native* pada perangkat, pada skripsi ini akan menggunakan Capacitor dibandingkan dengan Cordova. Capacitor mengelola *plugin* dengan cara yang berbeda dibandingkan dengan Cordova, yaitu dengan cara membangun semua *plugin* sebagai sebuah *libraries* di Android, dan diinstal menggunakan *management tool* android, yaitu Gradle. Capacitor didukung langsung oleh Ionic, dengan pengembangan yang lebih baru dibandingkan Cordova dapat lebih mendukung untuk *Web Apps* modern untuk membuka fungsionalitas *native* dari platform melalui API.

Dengan digunakannya Capacitor, maka akan digunakan Capacitor *plugin* diantaranya yaitu:

- Capacitor Browser API

Plugin ini berfungsi untuk menjalankan kemampuan *in-app browser* pada aplikasi WSDC 2017 Bali, yaitu untuk membuka berita terkait acara WSDC 2017 Bali. Untuk membuka *in-app browser*, digunakan *method* `open()` dengan properi url sebagai url yang dituju, seperti pada kode 3.42.

```

1 launch(newsUrl: string) {
2   Browser.open({ url: newsUrl });
3 }

```

Kode 3.42: *Method open()* Pada Browser API

- Capacitor Community Google Maps

Plugin ini berfungsi untuk menampilkan peta Google Maps secara *native* pada perangkat pengguna. Peta tersebut berisi lokasi dari seluruh acara WSDC 2017 Bali dilaksanakan yang ditandai dengan *marker*, dan juga menampilkan posisi dari perangkat pengguna. Seperti yang sudah dijelaskan pada sub bab 2.3.1.1, Capacitor Community Google Maps membutuhkan sebuah kontainer pada HTML untuk ditempati. Contoh kontainer tersebut seperti pada gambar 3.10a yang ditandai dengan kotak berwarna merah. Kode HTML untuk kontainer seperti pada kode 3.43 dibuat dengan ukuran lebar sesuai dengan layar ponsel pengguna, dan tinggi sebesar 400px. Kemudian mengisi *option* *boundingRect* pada *method* *createMap()* seperti pada kode 3.44 dengan nilai yang diambil dari kontainer pada HTML yang sudah dibuat sebelumnya. Dengan begitu, Google Maps sudah memiliki tempat di dalam kontainer pada HTML. Peta Google Maps menampilkan lokasi awal Kuta, Bali dengan memasukan *latitude* dan *longitude* dari Kuta, Bali ke dalam *option* *cameraPosition* properti target. Selain properti target, digunakan juga properti *zoom* yang digunakan untuk seberapa besar peta ditampilkan. Semakin besar nilai *zoom*, maka peta ditampilkan semakin dekat, begitu pula sebaliknya.

```

1 <div id="container">
2   <div id="mapContainer"></div>
3 </div>

```

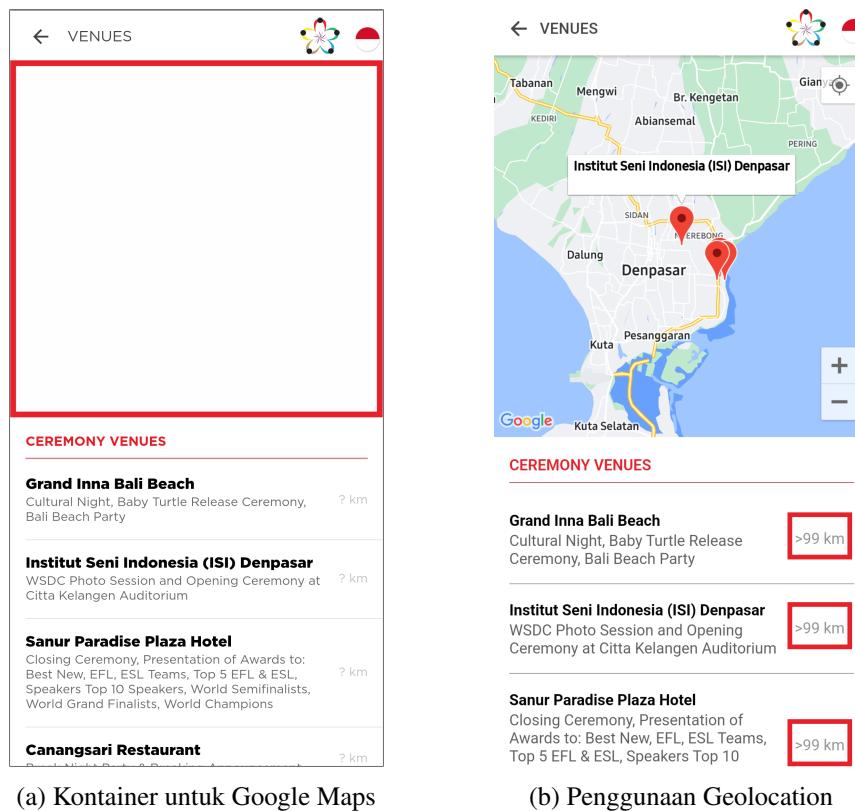
Kode 3.43: Kontainer pada HTML untuk Peta Capacitor CommunityGoogle Maps

```

1 const result = await CapacitorGoogleMaps.createMap({
2   boundingRect: {
3     width: Math.round(boundingRect.width),
4     height: Math.round(boundingRect.height),
5     x: Math.round(boundingRect.x),
6     y: Math.round(boundingRect.y),
7   },
8   cameraPosition: {
9     target: {
10       latitude: -8.722396,
11       longitude: 115.17671,
12     },
13     zoom: 11,
14   },
15   preferences: {
16     controls: {
17       isCompassButtonEnabled: true,
18       isMyLocationButtonEnabled: true,
19       isZoomButtonsEnabled: true,
20     },
21     appearance: {
22       isMyLocationDotShown: true,
23     },
24   },
25 });

```

Kode 3.44: *Method createMap()* Pada Capacitor Community Google Maps



Gambar 3.10: Penggunaan Community Google Maps dan Geolocation pada Aplikasi WSDC 2017 Bali

Pada Capacitor Community Google Maps terdapat *option* preferences seperti pada kode 3.44 yang digunakan untuk menambah fungsi pada Google Maps, diantaranya yaitu:

1. `isCompassButtonEnabled`: Digunakan untuk menampilkan tombol kompas sebagai petunjuk arah mata angin.
2. `isMyLocationButtonEnabled`: Digunakan untuk menampilkan tombol gps. Ketika tombol tersebut ditekan, posisi peta mengarah pada posisi ponsel pengguna. Penggunaan `isMyLocationButtonEnabled` adalah pada gambar 3.11a yang ditunjukan dengan kotak berwarna hijau.
3. `isZoomButtonsEnabled`: Digunakan untuk menampilkan tombol *zoom in* dan *zoom out*. Ketika tombol tersebut ditekan, maka peta akan melakukan *zoom in* dan *zoom out* sesuai dengan tombol yang ditekan. Penggunaan `isMyLocationButtonEnabled` adalah pada gambar 3.11a yang ditunjukan dengan kotak berwarna kuning.
4. `isMyLocationDotShown`: Digunakan untuk menampilkan titik biru sebagai tanda lokasi pengguna saat ini. Titik ini akan muncul ketika peta mengarah pada posisi ponsel pengguna. Penggunaan `isMyLocationButtonEnabled` adalah pada gambar 3.11b yang ditunjukan dengan kotak berwarna biru.

Di dalam peta Google Maps menampilkan *marker* untuk setiap lokasi *venue* seperti pada gambar 3.11a yand ditandai dengan kotak warna hitam. Untuk menampilkan *marker* digunakan *method* `addMarker()` seperti pada kode 3.45. *Marker* ditempatkan pada lokasi *venue* sesuai dengan koordinat *latitude* dan *longitude* masing-masing *venue*. Lokasi tersebut ditambahkan pada *option* `position`. Terdapat juga *option* preferences yang memiliki properti `title`, digunakan untuk menampilkan nama *venue* diatas *marker* ketika *marker* ditekan seperti pada gambar 3.11a yang ditandai dengan kotak berwarna merah.

```

1 CapacitorGoogleMaps.addMarker({
2   mapId: result.googleMap.mapId,
3   position: {
4     latitude: koor[1],
5     longitude: koor[0],
6   },

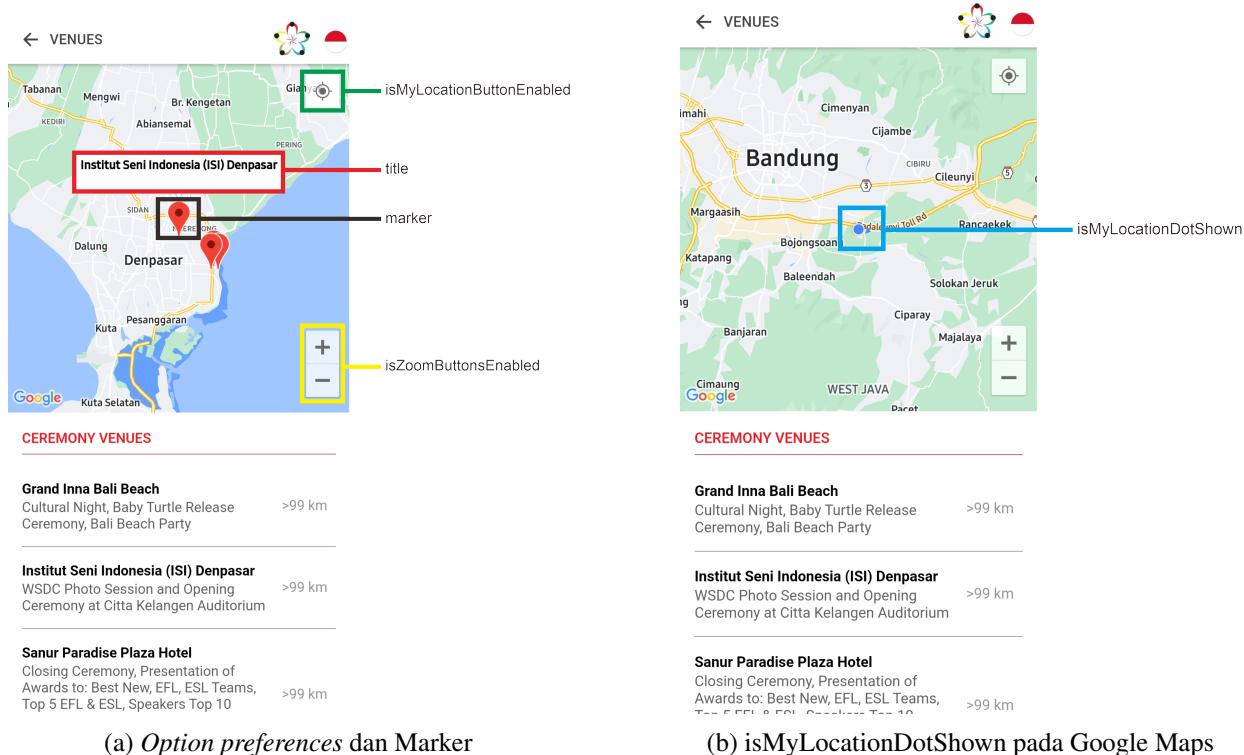
```

```

7     preferences: {
8         title: venuesMarker.properties.Name,
9     },
10 });

```

Kode 3.45: Method addMarker() Pada Capacitor Community Google Maps



Gambar 3.11: Penggunaan Community Google Maps pada Aplikasi WSDC 2017 Bali

- Capacitor Geolocation API

Capacitor Geolocation API digunakan untuk mengetahui posisi dari pengguna. *Plugin* ini mendapatkan koordinat berupa *latitude* dan *longitude* ponsel pengguna berada saat ini menggunakan *Global Positioning System (GPS)* dengan menggunakan *method* `getCurrentPosition()` seperti pada kode 3.46. Setelah mendapatkan koordinat ponsel pengguna, koordinat tersebut digunakan untuk menghitung jarak dari lokasi pengguna ke lokasi *venue* seperti pada gambar 3.10b yang ditandai dengan kotak berwarna merah. Jika lokasi pengguna dengan *venue* lebih dari 99 KM dari lokasi *venue*, maka akan menampilkan >99km. Jika lokasi pengguna dengan *venue* kurang dari 99 KM, maka akan menampilkan jarak yang sesuai.

```

1 const printCurrentPosition = async () => {
2     const coordinates = await Geolocation.getCurrentPosition();
3     this.userCoordinatesLat = coordinates.coords.latitude;
4     this.userCoordinatesLng = coordinates.coords.longitude;
5 };

```

Kode 3.46: Method getCurrentPosition(Pada Geolocation API

- Capacitor Splash Screen API

Capacitor Splash Screen API digunakan untuk menampilkan *splash screen* yang tampil pada saat aplikasi pertama kali dibuka. Seperti yang sudah dijelaskan pada sub-bab 2.3.1.1, Splash Screen secara *default* otomatis ditutup setelah 500ms sejak pertama kali Splash Screen dibuka. Namun jika hal itu terjadi, maka akan ada kemungkinan ketika Splash Screen ditutup, komponen-komponen pada aplikasi belum selesai dimuat, mengakibatkan pengguna melihat halaman putih kosong. Untuk menghindari

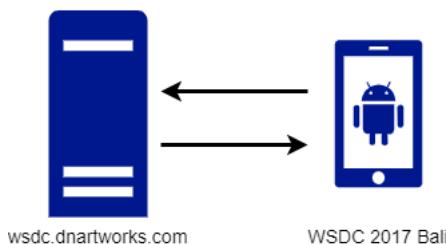
hal tersebut, Splash Screen ditutup ketika komponen Home, komponen yang pertama kali ditampilkan, selesai dimuat dan ditampilkan ke layar. Hal tersebut dapat terjadi dengan menjalankan *method* `hide()` yang dimiliki Capacitor Splash Screen pada *method* `ionViewDidEnter()` di dalam komponen Home seperti pada kode 3.47. *Method* `ionViewDidEnter()` sendiri merupakan sebuah Ionic Life Cycle yang dijalankan ketika komponen selesai dijalankan dan sudah ditampilkan ke layar.

```

1 ionViewDidEnter() {
2     SplashScreen.hide()
3 }
```

Kode 3.47: *Method* `hide()` Pada Splash Screen API

Pada sistem usulan, digunakan *libraries* yang dimiliki Angular, yaitu Angular Routing dan Angular HttpClient. Angular Routing digunakan untuk menangani navigasi dari satu tampilan ke tampilan lain, kemudian menampilkannya di layar. Angular HttpClient digunakan untuk melakukan komunikasi dengan server dengan menggunakan protokol HTTP.



Gambar 3.12: Arsitektur Komunikasi Server dengan Aplikasi WSDC 2017 Bali

Seperti pada Gambar 3.12, aplikasi WSDC 2017 Bali melakukan komunikasi dengan server yaitu `wsdc.dnartworks.com` menggunakan HttpClient. HttpClient melakukan sebuah *request* ke server berupa data JSON. Server kemudian menanggapi dan mengirimkan data yang diminta oleh HttpClient ke aplikasi WSDC 2017 Bali. Data tersebut digunakan untuk menampilkan data-data yang sesuai untuk masing-masing halaman pada aplikasi WSDC 2017 Bali.

Pada komponen Home dan Announcements HttpClient digunakan untuk mengambil data dari server seperti pada kode 3.48 yang merupakan penggunaan HttpClient pada komponen Home untuk sistem kini. Pertama, Ionic Storage mengambil data terlebih dahulu dari penyimpanan. Jika aplikasi pertama kali dijalankan setelah instalasi, maka data tersebut tidak akan ada di dalam penyimpanan. Jika data tidak ada data pada penyimpanan, HttpClient mengambil data dari asset lokal dengan *method* `get()`, dan menyimpannya ke dalam penyimpanan dengan Ionic Storage. Jika data tidak berhasil didapatkan, maka akan menampilkan sebuah Toast. Kemudian *method* `get()` dijalankan untuk mengambil data terbaru dari server. Data terbaru tersebut kemudian menggantikan data sebelumnya yang didapat dari lokal asset. Mengambil data pertama kali dari asset lokal bertujuan agar jika pengguna tidak terhubung ke koneksi internet pada saat pengguna melakukan instalasi aplikasi, aplikasi tetap dapat dibuka, dijalankan, dan menampilkan data-data yang seharusnya ditampilkan. Pengguna tetap mendapatkan data terbaru dari server ketika pengguna terhubung ke koneksi internet, karena *method* `get` pada HttpClient selalu dijalankan pada saat pengguna membuka aplikasi untuk selalu mendapatkan data terbaru dari server.

```

1 this.storage.get('wsdcDataStorage').then((data) => {
2     if(data == null){
3         this.http.get('../assets/json/wsdc_data.json').subscribe((data: any) => {
4             this.wsdcData = data;
5             this.storage.set('wsdcDataStorage',data);
6         },
7         error => {
8             this.showToast('Failed to refresh information from local storage');
9         });
10    }else{
```

```

11     this.wsdcData = data;
12 }
13 setTimeout(() => {
14     this.http.get('https://wsdc.dnartworks.com/wsdc_data.json')
15     .subscribe((data: any) => {
16         this.storage.set('wsdcDataStorage', data);
17         this.wsdcData = data;
18     },
19     error => {
20         this.showToast('Failed to refresh information');
21     });
22 }, 1000);
23 })

```

Kode 3.48: HTTPClient pada Komponen Home

Perubahan-perubahan yang terjadi dan dilakukan oleh penulis dari aplikasi WSDC 2017 Bali terdahulu ke aplikasi WSDC 2017 Bali terbaru meliputi perubahan UI Component, Native API dan penggunaan *plugin* dari Native API yaitu seperti pada tabel 3.9. Selain hal-hal yang berkaitan dengan perubahan-perubahan tersebut, pembuatan aplikasi WSDC 2017 Bali dengan Ionic 6 menggunakan asset yang sama dengan aplikasi WSDC 2017 Bali terdahulu, seperti gambar, file HTML, kelas TypeScript, file SCSS untuk setiap komponen, dan kode warna.

Tabel 3.9: Tabel Perubahan yang Dilakukan dari Ionic 3 ke Ionic 6

Jenis Perubahan	Perubahan	
	Ionic 3	Ionic 6
UI Component	<ion-item>	Menambahkan <ion-label>
	<ion-list-header>	Menambahkan <ion-label>
	<button>	<ion-button>
	<ion-segment-button>	Menambahkan <ion-label>
	<ion-scroll>	Dihapus, hanya menggunakan <ion-content> sudah cukup
Native API	Cordova	Capacitor
Plugin Native API	Cordova Google Maps API	Capacitor Community Google Maps API
	Cordova Geolocation	Capacitor Geolocation API
	Cordova Splash Screen	Capacitor Splash Screen API
	Cordova In-App Browser	Capacitor Browser API

3.2 Tantangan Pengembangan Sistem Usulan

Saat sedang melakukan proses migrasi aplikasi WSDC 2017 Bali dari Ionic Framework versi 3 ke Ionic Framework versi 6, terdapat beberapa kendala yang dialami. Kendala-kendala tersebut adalah sebagai berikut :

- Seperti yang disebutkan pada landasan teori (Sub Bab 2.3.3) sebelum melakukan migrasi dari Ionic Framework versi 3 ke Ionic Framework versi 6 terlebih dahulu melakukan migrasi dari Ionic Framework versi 3 ke Ionic Framework versi 4, yang selanjutnya dari Ionic versi 4 ke Ionic versi 5 dan dari Ionic versi 5 ke Ionic versi 6. Namun karena tidak tersedianya perintah untuk membuat aplikasi dengan menggunakan Ionic Framework versi 4 dan 5, maka penulis langsung melakukan migrasi dari Ionic Framework versi 3 ke Ionic Framework versi 6. Dalam melakukan hal ini, penulis berlandaskan bahwa susunan kelas Ionic Framework versi 4, 5 dan 6 tidaklah berubah sama sekali. Yang mengalami perubahan hanyalah pembaruan properti mengenai API, CSS, dan *package dependencies* yang terpasang, yang telah dijelaskan pada landasan teori (Sub Bab 2.3.3).
- Pada awal pengerjaan skripsi, halaman Draw dan Result pada aplikasi WSDC 2017 Bali tidak dapat diakses karena terjadi kesalahan konfigurasi pada server. Kemudian setelah menghubungi dan dibantu

oleh pembuat dari aplikasi WSDC 2017 Bali, maka masalah ini telah terselesaikan, yaitu halaman Draw dan Result pada aplikasi WSDC 2017 Bali dapat diakses kembali sebagaimana mestinya.

- Pada saat penggerjaan skripsi ini, pada Desember 2021 Ionic meluncurkan generasi terbaru dari Ionic Framework, yaitu Ionic versi 6. Sedangkan Ionic versi 5 pengembangannya berhenti didukung pada tanggal 8 Juni 2022¹. Maka dari itu, atas saran dan masukan dosen pembimbing dan dosen pengaji, maka peneliti melakukan migrasi tidak lagi sampai Ionic versi 5, melainkan sampai dengan Ionic versi 6 untuk masa dukungan Ionic Framework yang lebih lama.
- Pada Capacitor Community Google Maps, terdapat *option* preferences isCompassButtonEnabled yang digunakan untuk menampilkan tombol kompas pada peta. Namun setelah aplikasi dijalankan, tombol tersebut tidak muncul pada layar. Maka hal ini menyebabkan adanya perbedaan tampilan pada peta antara aplikasi WSDC 2017 Bali terdahulu dan terbaru, yaitu tidak ada tombol kompas pada aplikasi WSDC 2017 Bali terbaru.

¹Status pemeliharaan dan dukungan dari setiap versi Ionic dilihat pada <https://ionicframework.com/docs/reference/support>

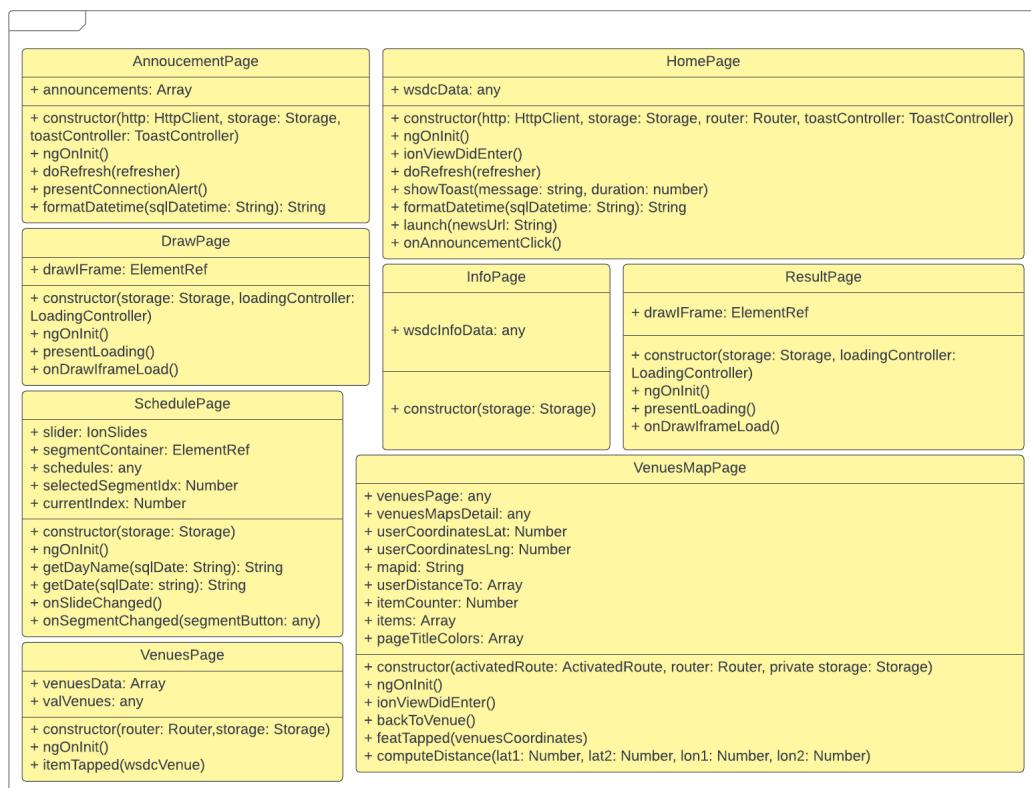
BAB 4

PERANCANGAN

Pada bab ini menjelaskan mengenai perancangan aplikasi yang akan dibangun meliputi perancangan kelas pada masing-masing komponen beserta dengan deskripsi dan fungsinya, dan perancangan struktur HTML. Untuk antarmuka pada aplikasi yang dibangun memiliki antarmuka yang sama dengan aplikasi WSDC 2017 Bali terdahulu. Penjelasan terkait dengan antarmuka telah dibahas pada bagian 3.1.

4.1 Perancangan Kelas

Pada aplikasi WSDC 2017 Bali yang akan dibangun menggunakan struktur kelas yang sama dengan aplikasi WSDC 2017 Bali terdahulu, dengan beberapa penyesuaian terkait dengan pembaruan yang dilakukan. Diagram kelas secara keseluruhan dapat dilihat pada gambar 4.1.



Gambar 4.1: Diagram Kelas Keseluruhan

Perancangan kelas dari masing-masing komponen adalah sebagai berikut:

1. Komponen *Announcements*

Di dalam komponen *announcements* terdapat sebuah kelas *AnnouncementPage*. Kelas ini berfungsi untuk mengambil data *announcements* dari *storage*, dan menampilkan pengumuman ke halaman *announcements*. Kelas ini memiliki sebuah atribut, yaitu *announcements* bertipe *array* yang menyimpan localtime bertipe *string* yang berisi tanggal dan waktu dari sebuah pengumuman dikeluarkan dengan format “Tahun-Bulan-Hari Jam:Menit:Detik”, dan message yang berisi pesan pengumuman yang bertipe data *string*. Kelas ini memiliki beberapa *method*, diantaranya adalah sebagai berikut:

- **constructor(private http: HttpClient, private storage: Storage, public toastController: ToastController)**

Method constructor berfungsi untuk memberikan nilai awal pada saat kelas dibuat. **Parameter:**

- **http:** Parameter ini digunakan untuk menyimpan API HttpClient.
- **storage:** Parameter ini digunakan untuk menyimpan API Storage.
- **toastController:** Parameter ini digunakan untuk menyimpan API ToastController.

Kembalian: tidak ada.

- **ngOnInit()**

Method ini berfungsi untuk mengambil data *announcements* yang terdapat di dalam *storage*, kemudian menyimpannya ke dalam atribut kelas, yaitu *announcements*.

Parameter: tidak ada.

Kembalian: tidak ada.

- **doRefresh(refresher)**

Method ini berfungsi untuk melakukan penyegaran ulang terhadap halaman *Announcements*. Saat melakukan penyegaran ulang, *method* ini mengambil data terbaru dari server, kemudian menyimpannya ke dalam atribut kelas, yaitu *announcements*. Selanjutnya akan dilakukan penghapusan terlebih dahulu terhadap data yang sudah ada di dalam *storage*, kemudian menyimpan data terbaru yang di dapatkan dari server ke dalam *storage*. Jika server tidak merespon dan data tidak dapat diambil, maka akan memanggil *method* presentConnectionAlert() untuk menampilkan *toast*.

Parameter: *Method* ini memiliki sebuah parameter yaitu *refresher*, yang berisi *event refresher*.

Kembalian: tidak ada.

- **presentConnectionAlert()**

Method ini berfungsi untuk menampilkan *toast* yang akan menampilkan sebuah tulisan “Failed to refresh information” selama tiga detik. *Method* ini hanya akan digunakan ketika *method* doRefresh tidak berhasil mengambil data terbaru dari server.

Parameter: tidak ada.

Kembalian: tidak ada.

- **formatDatetime(sqlDatetime: string)**

Method ini berfungsi untuk mengambil jam, menit, hari, dan bulan dari parameter.

Parameter: *sqlDatetime*: detail waktu dan tanggal pada sebuah pengumuman.

Kembalian: sebuah *string* yang berisi tanggal dan waktu dari sebuah pengumuman dengan format “Jam-Menit | Hari, Tanggal, Bulan”.

2. Komponen *Draw*

Di dalam komponen *announcements* terdapat sebuah kelas *DrawPage*. Kelas ini berfungsi untuk mengambil data *draw* dari *storage*, serta menampilkan data *draw* ke halaman *draw*. Kelas ini memiliki sebuah atribut, yaitu *drawIFrame* yang bertipe *ElementRef*. Atribut ini merupakan sebuah *ViewChild* yang digunakan untuk memanggil elemen dari komponen *draw*, yaitu *drawIFrame* pada file *draw.page.html*.

Kelas ini memiliki beberapa *method*, diantaranya adalah sebagai berikut:

- **constructor(private storage: Storage,public loadingController: LoadingController)**
Method constructor berfungsi untuk memberikan nilai awal pada saat kelas dibuat. **Parameter:**
 - **storage:** Parameter ini digunakan untuk menyimpan API Storage.
 - **loadingController:** Parameter ini digunakan untuk menyimpan API LoadingController.**Kembalian:** tidak ada.
- **ngOnInit()**
Method ini berfungsi untuk mengambil data *draws* yang terdapat di dalam *storage*. Data tersebut lalu disimpan ke dalam *child* drawIFrame. Kemudian *method* ini akan memanggil *method* presentLoading().
Parameter: tidak ada.
Kembalian: tidak ada.
- **async presentLoading()**
Method ini berfungsi untuk menampilkan sebuah indikator *loading* dengan pesan “Please wait...”.
Parameter: tidak ada.
Kembalian: tidak ada.
- **onDrawIframeLoad()**
Method ini merupakan sebuah *template statement* yang dipanggil oleh *event* di dalam *tag <iframe>* pada *file* draw.page.html. *Method* ini berfungsi untuk menampilkan data *draw* yang disimpan di dalam *storage*.
Parameter: tidak ada.
Kembalian: tidak ada.

3. Komponen Home

Di dalam komponen *home* terdapat sebuah kelas HomePage. Kelas ini menjadi sebagai kelas yang pertama kali diakses oleh aplikasi. Maka dari itu, kelas ini berfungsi untuk menginisialisasi sebuah *storage* yang diisi oleh data yang digunakan oleh seluruh kelas pada aplikasi. Kelas ini memiliki sebuah atribut, yaitu wsdcData yang bertipe any. Atribut ini akan menyimpan data json berisi data yang akan digunakan untuk aplikasi.

Kelas ini memiliki beberapa *method*, diantaranya adalah sebagai berikut:

- **constructor(private http: HttpClient, private storage: Storage, private router: Router, public toastController: ToastController)**
Method constructor berfungsi untuk memberikan nilai awal pada saat kelas dibuat. **Parameter:**
 - **http:** Parameter ini digunakan untuk menyimpan API HttpClient.
 - **storage:** Parameter ini digunakan untuk menyimpan API Storage.
 - **router:** Parameter ini digunakan untuk menyimpan API Router.
 - **toastController:** Parameter ini digunakan untuk menyimpan API ToastController.**Kembalian:** tidak ada.
- **ionViewDidEnter()**
Method ini dijalankan ketika halaman sudah masuk sepenuhnya. *Method* ini berfungsi untuk menutup *splash screen*.
Parameter: tidak ada.
Kembalian: tidak ada.
- **ngOnInit()**
Method ini berfungsi untuk mengambil data json dari *storage*. Jika data di dalam *storage* tersebut belum pernah dibuat sebelumnya, maka *method* ini akan membuat sebuah data baru di dalam *storage* yang berisi data json yang dibutuhkan untuk aplikasi. Secara *default*, untuk mengantisipasi jika pengguna tidak memiliki koneksi internet, maka pertama kali *method* ini mengambil data adalah dari aset yang berada di *folder assets*. Setelah itu, *method* ini akan mengambil data terbaru dari server. Jika server tidak merespon dan data tidak berhasil didapatkan, maka akan memanggil *method* showToast().
Parameter: tidak ada.

Kembalian: tidak ada.

- **doRefresh(refresher)**

Method ini berfungsi untuk melakukan penyegaran ulang terhadap halaman *Home*. Saat melakukan penyegaran ulang, *method* ini akan mengambil data terbaru dari server, kemudian menyimpannya ke dalam atribut kelas, yaitu *wsdcData*. Selanjutnya akan dilakukan penghapusan terlebih dahulu terhadap data yang sudah ada di dalam *storage*, kemudian menyimpan data terbaru yang di dapatkan dari server ke dalam *storage*. Jika server tidak memberi respon dan data tidak dapat diambil, maka akan memanggil *method* *showToast()* untuk menampilkan *toast*.

Parameter: tidak ada.

Kembalian: tidak ada.

- **async showToast(message: string, duration: number=3000)**

Method ini berfungsi untuk menampilkan *toast* yang akan menampilkan pesan dan durasi sesuai dengan yang ada pada parameter.

Parameter:

- **message:** pesan yang akan ditampilkan di dalam *toast*.
- **duration:** durasi seberapa lama *toast* berada di layar.

Kembalian: tidak ada.

- **formatDatetime(sqlDatetime: string)**

Method ini berfungsi untuk mengambil jam, menit, hari, dan bulan dari parameter.

Parameter: *sqlDatetime*: detail waktu dan tanggal pada sebuah pengumuman.

Kembalian: sebuah *string* yang berisi tanggal dan waktu dari sebuah pengumuman dengan format “Hari | Jam-Menit”.

- **launch(newsUrl: string)**

Method ini berfungsi untuk membuka url berita sesuai dengan url yang ada di dalam parameter.

Parameter: *newsUrl*: *string* url dari sebuah berita.

Kembalian: tidak ada.

- **onAnnouncementClick()**

Method ini berfungsi untuk berpindah halaman ke halaman *announcements*.

Parameter: tidak ada.

Kembalian: tidak ada.

4. Komponen Info

Di dalam komponen info terdapat sebuah kelas *InfoPage*. Kelas ini berfungsi untuk mengambil data info dari *storage* dan menyimpannya ke atribut kelas yang kemudian data tersebut akan ditampilkan ke halaman info. Kelas ini memiliki sebuah atribut, yaitu *wsdcInfoData* yang bertipe *any*. Atribut ini digunakan untuk menyimpan data untuk halaman info. Kelas ini hanya memiliki sebuah *method* yaitu *method constructor*. *Method* ini memiliki sebuah parameter, yaitu *storage* yang bertipe *Storage*. Constructor pada kelas ini bertujuan untuk mengambil data info dari *storage* dan menyimpannya ke dalam atribut kelas, yaitu *wsdcInfoData*.

5. Komponen Result

Di dalam komponen *result* terdapat sebuah kelas *ResultPage*. Kelas ini berfungsi untuk mengambil data *result* dari *storage* yang kemudian data tersebut akan ditampilkan ke halaman *result*. Kelas ini memiliki sebuah atribut yaitu *resultIFrame* yang bertipe *ElementRef*. Atribut ini merupakan sebuah *@ViewChild* yang digunakan untuk memanggil elemen dari komponen *result*, yaitu *resultIFrame* pada file *result.page.html*.

Kelas ini memiliki beberapa *method*, diantaranya adalah sebagai berikut:

- **constructor(private storage: Storage, public loadingController: LoadingController)**

Method constructor berfungsi untuk memberikan nilai awal pada saat kelas dibuat. **Parameter:**

- **storage:** Parameter ini digunakan untuk menyimpan API *Storage*.
- **loadingController:** Parameter ini digunakan untuk menyimpan API *LoadingController*.

Kembalian: tidak ada.

- **ngOnInit()**

Method ini berfungsi untuk mengambil data *result* yang terdapat di dalam *storage*. Data tersebut lalu disimpan ke dalam *child resultIFrame*. Setelah itu *method* ini akan memanggil *method presentLoading()*.

Parameter: tidak ada.

Kembalian: tidak ada.

- **async presentLoading()**

Method ini berfungsi untuk menampilkan sebuah indikator *loading* dengan pesan “Please wait...”.

Parameter: tidak ada.

Kembalian: tidak ada.

- **onResultIframeLoad()**

Method ini merupakan sebuah *template statement* yang dipanggil oleh *event* di dalam tag <iframe> pada file *result.page.html*. *Method* ini berfungsi untuk menampilkan data *result* yang disimpan di dalam *storage*.

Parameter: tidak ada.

Kembalian: tidak ada.

6. Komponen *Schedule*

Di dalam komponen *schedule* terdapat sebuah kelas *SchedulePage*. Kelas ini berfungsi untuk mengatur jadwal pada halaman *schedule*, seperti mengatur perpindahan *slides* dan *segment* pada jadwal.

Kelas ini memiliki beberapa atribut, diantaranya adalah sebagai berikut:

- **slider:** Atribut ini merupakan sebuah @ViewChild yang digunakan untuk memanggil elemen dari komponen *schedule*, yaitu *scheduleSlider* pada file *schedule.page.html*.
- **segmentContainer:** Atribut ini merupakan sebuah @ViewChild yang digunakan untuk memanggil elemen dari komponen *schedule*, yaitu *segmentContainer* pada file *schedule.page.html*.
- **schedules:** Atribut ini akan menyimpan data *schedules* yang diambil dari *storage*.
- **slideOpts:** Atribut ini berisi pengaturan dasar untuk *slides*. Berisi *initialSlide* untuk mengatur *slides* ke berapa saat pertama kali halaman *schedule* dibuka, dan *speed* untuk mengatur kecepatan transisi antar *slides*.
- **selectedSegmentIdx:** Atribut ini digunakan untuk menyimpan index dari *segment* yang sedang aktif.
- **currentIndex:** Atribut ini digunakan untuk menyimpan index dari *slides* yang sedang aktif.

Kelas ini memiliki beberapa *method*, diantaranya adalah sebagai berikut:

- **constructor(private storage: Storage)**

Method constructor berfungsi untuk memberikan nilai awal pada saat kelas dibuat.

Parameter: storage: Parameter ini digunakan untuk menyimpan API Storage.

Kembalian: tidak ada.

- **ngOnInit()**

Method ini berfungsi untuk mengambil data *schedule* yang terdapat di dalam *storage*. Data tersebut lalu disimpan ke dalam atribut *schedules*.

Parameter: tidak ada.

Kembalian: tidak ada.

- **getDayName(sqlDate: string)**

Method ini berfungsi untuk mengambil hari dari parameter.

Parameter: sqlDate: Sebuah *string* yang berisi tahun, bulan, dan tanggal.

Kembalian: *string* nama hari.

- **getDate(sqlDate: string)**

Method ini berfungsi untuk mengambil tanggal dari parameter.

Parameter: sqlDate: Sebuah *string* yang berisi tahun, bulan, dan tanggal.

Kembalian: *string* tanggal.

- **onSlideChanged()**

Method ini dipanggil saat *slides* dipindahkan dengan cara digeser ke kanan atau ke kiri. *Method* ini akan mengubah atribut *currentIndex* menjadi index *slides* saat ini, kemudian mengubah atribut

`selectedSegmentIdx` menjadi index `slides` saat ini. Hal ini bertujuan agar indeks dari `segment` yang aktif dapat diganti sesuai dengan indeks `slides` yang aktif. Dengan begitu tampilan `segment` dan `slides` yang aktif akan sesuai.

Parameter: tidak ada.

Kembalian: tidak ada.

- **onSegmentChanged(segmentButton)**

Method ini berfungsi untuk mengubah `slides` yang aktif sesuai dengan indeks dari `segment` yang sedang aktif.

Parameter: `segmentButton`: Merupakan sebuah *event* dari `segment` yang akan diambil `value` yang berisi indeks dari `segment` yang aktif.

Kembalian: tidak ada.

7. Komponen *Venues*

Di dalam komponen *venues* terdapat sebuah kelas `VenuesPage`. Kelas ini berfungsi untuk mengambil data *venues* dari *storage* dan menyimpannya ke dalam atribut kelas. Kelas ini juga berfungsi melakukan navigasi ke halaman *venues map*.

Kelas ini memiliki beberapa atribut, yaitu:

- **venuesData**: Atribut ini bertipe array. Atribut ini digunakan untuk menyimpan data dari *venues* yang berisi id, name, icon, geojson, dan colorIdx dari masing masing kategori *venues*. Data ini nantinya akan dikirimkan ke kelas *Venues Maps*.
- **valVenues**: Digunakan untuk menyimpan data *venues* yang didapatkan dari *storage*.

Kelas ini memiliki beberapa *method*, diantaranya yaitu:

- **constructor(private router: Router,private storage: Storage)**

Method constructor berfungsi untuk memberikan nilai awal pada saat kelas dibuat.

Parameter:

- **router**: Parameter ini berfungsi untuk menyimpan API Router.
- **storage**: Parameter ini berfungsi untuk menyimpan API Storage.

Kembalian: tidak ada.

- **ngOnInit()**

Method ini berfungsi untuk mengambil data *venues* yang terdapat di dalam *storage*. Data tersebut lalu disimpan ke dalam atribut `valVenues`.

Parameter: tidak ada.

Kembalian: tidak ada.

- **itemTapped(wsdcVenue)**

Method ini berfungsi untuk melakukan navigasi ke halaman *Venues Maps* dengan bantuan Router milik Angular. *Method* ini akan mengirimkan sebuah data array yang didapatkan dari parameter ke halaman *Venues Maps*.

Parameter: `wsdcVenue`: Parameter ini merupakan sebuah array yang berisi sama seperti atribut `venuesData`. Isi dari array ini adalah data untuk sebuah *venues* yang ingin dilihat.

Kembalian: tidak ada.

8. Komponen *Venues Maps*

Di dalam komponen *venues Maps* terdapat sebuah kelas `VenuesMapsPage`. Kelas ini berfungsi untuk mengambil data *venues* yang dikirimkan dari komponen *Venues*, menampilkan peta *venues*, serta melakukan kalkulasi jarak pengguna dengan *venues*. Kelas ini memiliki beberapa atribut, yaitu:

- **venuesPage** : Atribut ini digunakan untuk menyimpan data yang dikirimkan dari komponen *Venues*.
- **venuesMapsDetail** : Atribut ini digunakan untuk menyimpan data *venues* dari *storage* sesuai dengan kategori yang sedang dipilih.
- **userCoordinatesLat** : Atirbut ini digunakan untuk menyimpan koordinat latitude dari perangkat pengguna.
- **userCoordinatesLng** : Atirbut ini digunakan untuk menyimpan koordinat longitude dari perangkat pengguna.

- **mapid** : Atribut ini digunakan untuk menyimpan id dari peta.
- **userDistanceTo** : Atribut ini digunakan untuk menyimpan jarak dari posisi perangkat pengguna ke posisi *venues*.
- **itemCounter** : Atribut ini digunakan untuk menyimpan berapa banyak *venues* dalam sebuah kategori.
- **items** : Atribut ini digunakan untuk menyimpan id dan id warna dari sebuah kategori *venues*.
- **pageTitleColors** : Atribut ini merupakan sebuah *array* yang berisi kode warna untuk masing-masing judul kategori *venues*.

Kelas ini memiliki beberapa *method*, diantaranya adalah sebagai berikut:

- **constructor(private activatedRoute: ActivatedRoute, private router: Router, private storage: Storage)**

Method constructor berfungsi untuk memberikan nilai awal pada saat kelas dibuat. *Constructor* juga digunakan untuk mengambil data dari *storage* dan memasukannya ke atribut *venuesMapsDetail*.

Parameter:

- **activatedRoute** : Parameter ini digunakan untuk menyimpan API *ActivatedRoute*.
- **router** : Parameter ini digunakan untuk menyimpan API *Router*.
- **storage** : Parameter ini digunakan untuk menyimpan API *Storage*.

Kembalian: tidak ada.

- **ngOnInit()**

Method ini berfungsi untuk mengambil data *venues* yang terdapat di dalam *storage*. Data tersebut lalu disimpan ke dalam atribut *venuesMapsDetail*.

Parameter: tidak ada.

Kembalian: tidak ada.

- **ionViewDidEnter()**

Method ini dijalankan ketika halaman sudah masuk sepenuhnya. *Method* ini digunakan untuk menginisialisasi peta menggunakan *plugin* Google Maps yang disediakan oleh Capacitor. Peta tersebut menampilkan Pulau Bali, lebih tepatnya di Kecamatan Kuta dengan latitude -8.722396 dan longitude 115.17671. *Method* ini juga membuat *marker* yang menandai setiap lokasi *venues* pada satu kategori *venues*. *Method* ini juga digunakan untuk menyimpan jarak antara posisi perangkat pengguna ke masing-masing posisi *venues*, yang kemudian disimpan ke dalam atribut *userDistanceTo*.

Parameter: tidak ada.

Kembalian: tidak ada.

- **backToVenue()**

Method ini digunakan untuk bernavigasi kembali ke halaman *Venues*.

Parameter: tidak ada.

Kembalian: tidak ada.

- **featTapped(venuesCoordinates)**

Method ini digunakan untuk mengarahkan kamera map ke arah lokasi *venues* yang dituju sesuai dengan koordinat pada parameter.

Parameter: *venuesCoordinates*: Parameter ini berisi koordinat latitude dan longitude dari lokasi *venues* yang dituju.

Kembalian: tidak ada.

- **computeDistance(lat1, lat2, lon1, lon2)**

Method ini berfungsi untuk menghitung jarak dari posisi perangkat pengguna ke posisi *venues* menggunakan latitude dan longitude dari kedua posisi tersebut.

Parameter:

- **lat1**: koordinat latitude dari pengguna.
- **lat2**: koordinat latitude dari *venues*.

- **lon1**: koordinat longitude dari pengguna.
- **lon2**: koordinat longitude dari *venues*.

Kembalian: *String* jarak dari posisi perangkat pengguna ke posisi *venues*.

4.2 Perancangan Struktur HTML

Struktur HTML pada masing-masing komponen mengambil struktur yang sama dengan aplikasi WSDC 2017 Bali terdahulu, namun dengan beberapa perubahan. Perubahan-perubahan tersebut telah dibahas pada bagian 2.3.3, serta analisis penggunaannya di sistem usulan pada bagian 3.1.

Masing-masing HTML yang terdapat pada setiap komponen memiliki struktur yang sama, yaitu terdapat sebuah *header* dan sebuah *content*. Penjelasan struktur pada *header* dan *content* adalah sebagai berikut:

1. Header

Header untuk setiap komponen pada umumnya memiliki struktur yang serupa namun dibedakan dengan judul dari setiap halaman. *Header* digunakan untuk menampilkan judul dari sebuah halaman, serta menyediakan sebuah *menu button* sebagai salah satu cara untuk membuat *sidemenu* untuk melakukan navigasi antar halaman. *Header* dibungkus oleh tag `<ion-header>` yang didalamnya terdapat tag `<ion-toolbar>` yang disediakan Ionic Framework. Kemudian untuk *menu button* dibuat oleh tag `<ion-menu-button>` pada tag `<ion-buttons>`. Selanjutnya untuk judul dari sebuah halaman dibungkus oleh tag `<ion-title>`. Judul akan berbeda beda tergantung dengan halamannya. Salah satu contoh dari penggunaan *header* adalah pada gambar 3.4a yang ditandai dengan kotak berwarna biru.

2. Content

Content untuk setiap komponen memiliki struktur yang berbeda-beda tergantung dengan isi dari halaman tersebut. Namun *content* untuk setiap komponen dibungkus oleh sebuah tag `<ion-content>`. Salah satu contoh dari penggunaan *content* adalah pada gambar 3.4a yang ditandai dengan warna merah. Struktur *content* untuk masing-masing halaman adalah sebagai berikut:

- Halaman *Announcements*

Content pada halaman *announcements* berisi sebuah *refresher* dengan tag `<ion-refresher>` untuk melakukan penyegaran ulang terhadap halaman *announcements* yaitu mengambil data terbaru dari server. Penggunaan tag `<ion-refresher>` seperti pada gambar 3.2a yang ditandai dengan kotak berwarna hijau. Terdapat sebuah list dengan tag `<ion-list>` yang ditandai dengan kotak berwarna kuning. List menampilkan pengumuman yang berisi waktu dan tanggal, serta pesan dari pengumuman tersebut. Setiap satu pengumuman dibungkus oleh sebuah tag `<ion-item>` yang ditandai dengan kotak berwarna hitam. Di dalamnya terdapat sebuah tag `<ion-label>` yang berisi tag `<h3>` untuk waktu dan tanggal, serta tag `<p>` untuk pesan pengumuman.

- Halaman *Draw*

Content pada halaman *draw* berisi sebuah tag `<iframe>` yang digunakan untuk menyematkan dokumen lain ke dalam dokumen HTML, yaitu sebuah data *draw* yang berisi pembagian grup proposisi dan oposisi bagi setiap negara peserta WSDC 2017 Bali. Data tersebut diambil dari server, kemudian dimasukan ke dalam tag `<iframe>`. Penggunaan tag `<iframe>` seperti pada gambar 3.3a yang ditandai dengan kotak berwarna hijau.

- Halaman *Home*

Content pada halaman *home* berisi sebuah *refresher* dengan tag `<ion-refresher>` untuk melakukan penyegaran ulang terhadap halaman *home*, yaitu mengambil data terbaru dari server. Penggunaan tag `<ion-refresher>` seperti pada gambar 3.4a yang ditandai dengan kotak berwarna hijau. Terdapat sebuah *card* dengan tag `<ion-card>` seperti yang ditandai dengan kotak berwarna merah muda yang digunakan untuk menampilkan sebuah pengumuman terbaru, berikut dengan waktu dan tanggal serta pesan dari pengumuman tersebut. Di dalam *card* terdapat *grid* untuk *layout* dari *card*. Di dalam sebuah *grid* terdapat sebuah baris dengan tag `<ion-row>`. Di dalam baris tersebut terdapat dua buah kolom dengan tag `<ion-col>`. Masing-masing

kolom memiliki ukuran, kolom pertama yang ditandai dengan kotak berwarna coklat berukuran sembilan digunakan untuk menyimpan *header* dari *card* dengan *title* yaitu “Latest Announcement”. *Header* ini dibungkus di dalam tag `<ion-card-header>`, dan *title* dengan tag `<ion-card-title>`. Selain *header* untuk *card*, terdapat pula *content* untuk *card* dengan tag `<ion-card-content>` yang berisi waktu dan tanggal, serta pesan dari pengumuman. Untuk kolom selanjutnya dengan ukuran tiga berisi sebuah gambar seperti yang ditandai dengan kotak berwarna jingga.

Selain *card* pengumuman, terdapat juga list dengan tag `<ion-list>` yang berisi *thumbnail* dari berita-berita terkait acara WSDC 2017 Bali seperti yang ditandai dengan kotak berwarna ungu pada gambar 3.4a. Di dalam list terdapat sebuah *header* dengan tag `<ion-list-header>` yang berisi judul dari list yaitu “Newsletters” yang berada di dalam tag `<ion-label>`. Untuk masing-masing *thumbnail* berita berada di dalam tag `<ion-item>`. Untuk masing-masing item terdapat sebuah gambar *thumbnail* dari sebuah berita, judul dari berita tersebut, serta sebuah tombol dengan tag `<ion-button>` yang jika ditekan akan mengarahkan pengguna untuk melihat berita tertentu sesuai dengan item yang dipilih.

- Halaman Info

Content pada halaman info berisi sebuah *grid* dengan tag `<ion-grid>`, yang berisi sebuah baris dengan tag `<ion-row>`. Baris ini menyimpan info-info seputar kontak-kontak penting yang dapat dihubungi, kosa kata dalam Bahasa Indonesia sehari-hari, serta credits kepada pembuat aplikasi WSDC 2017 Bali.

- Halaman *Result*

Content pada halaman *result* berisi sebuah tag `<iframe>` yang digunakan untuk menyematkan dokumen lain ke dalam dokumen HTML, yaitu sebuah data *result* hasil dari keseluruhan pertandingan WSDC 2017 Bali. Data tersebut diambil dari server, kemudian dimasukan ke dalam tag `<iframe>`.

- Halaman *Schedule*

Content pada halaman *schedule* berisi *segment* dengan tag `<ion-segment>` dan sebuah *slides* dengan tag `<ion-slides>`. *Segment* digunakan untuk menampilkan tanggal dan hari, serta berfungsi untuk memindahkan *slides* ke hari yang dipilih seperti yang ditandai dengan kotak berwarna hijau pada gambar 3.7a. Untuk melakukan hal tersebut, di dalam *segment* terdapat sebuah *button* dengan tag `<ion-segment-button>` yang berisi tag `<ion-label>` untuk menampung tanggal dan hari. Sedangkan *slides* digunakan untuk menampilkan jadwal acara WSDC 2017 Bali pada hari yang sesuai dengan *segment* yang terpilih seperti yang ditandai dengan kotak berwarna coklat. Untuk menampilkan jadwal menggunakan *list* dengan tag `<ion-list>` yang ditandai dengan warna merah muda. Di dalam *list* terdapat sebuah *item* dengan tag `<ion-item>`. Untuk setiap *item* berisi waktu mulai dan selesai sebuah acara dengan tag `<ion-note>` yang ditandai dengan warna ungu, serta nama acara dengan tag `<h3>` yang ditandai dengan kotak berwarna jingga dan lokasi acara dengan tag `<p>` yang ditandai dengan kotak berwarna biru muda. Nama dan lokasi acara dibungkus dengan tag `<ion-label>`.

- Halaman *Venues*

Content pada halaman *venues* berisi *grid* dengan tag `<ion-grid>` dengan satu baris menggunakan tag `<ion-row>`. Di dalamnya terdapat sebuah *list* dengan tag `<ion-list>`. *List* tersebut berisi tombol dengan tag `<ion-button>` yang ditandai dengan kotak berwarna hijau pada gambar 3.8a. Masing-masing tombol digunakan untuk melakukan navigasi ke halaman *venues map*. Setiap tombol berisi ikon dengan tag `<ion-icon>` yang ditandai dengan kotak berwarna biru muda pada gambar 3.8a dan sebuah tag `` yang berisi nama dari *venues*, ditandai dengan kotak berwarna hitam.

- Halaman *Venues Map*

Content pada halaman *venues map* berisi tag `<div>` yang digunakan untuk menampung peta dari sebuah kategori *venues* seperti yang ditandai dengan kotak berwarna hijau pada gambar 3.9a. Terdapat sebuah label dengan tag `<ion-label>` yang berisi nama kategori *venues* yang ditandai

dengan kotak berwarna kuning. Kemudian untuk menampilkan nama dan deskripsi sebuah *venues*, serta jarak antara pengguna dengan *venues*, menggunakan *list* dengan *tag* `<ion-list>` yang ditandai dengan kotak berwarna biru muda.

BAB 5

IMPLEMENTASI DAN PENGUJIAN

Pada bab ini menjelaskan mengenai implementasi perangkat lunak, dan pengujian perangkat lunak. Implementasi perangkat lunak berisi penjelasan lingkungan pengembangan perangkat lunak dan hasil implementasi. Sedangkan pengujian perangkat lunak berisi hasil pengujian fungsional dan eksperimental terhadap perangkat lunak yang telah dibangun.

5.1 Implementasi

5.1.1 Lingkungan Implementasi

Implementasi perangkat lunak ini dilakukan di komputer penulis dengan spesifikasi berikut:

1. Sistem Operasi: Windows 10 version 21H2
2. Versi Android Development Kit (SDK): API 30 (Android 11 (R))
3. Versi Ionic CLI: 6.20.1
4. Versi Capacitor: 3.4.3

5.1.2 Hasil Implementasi

Hasil implementasi berupa sebuah aplikasi android WSDC 2017 Bali. Sebelum halaman dimuat, ditampilkan sebuah *splash screen* terlebih dahulu yang menampilkan logo WSDC, logo WSDC 2017 Bali, dan logo Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi Republik Indonesia. Tangkapan layar *splash screen* dapat dilihat pada Gambar 5.1a. Aplikasi WSDC 2017 Bali terdiri dari 8 halaman yang dapat diakses melalui *sidemenu*. Tangkapan layar *sidemenu* dapat dilihat pada Gambar 5.2a. Halaman-halaman yang ada pada aplikasi WSDC 2017 Bali tersebut yaitu:

1. Halaman Home

Halaman *home* menjadi halaman pertama yang dimasuki oleh pengguna di aplikasi WSDC 2017 Bali. Pada halaman ini pengguna dapat melihat pengumuman terbaru terkait dengan acara WSDC 2017 Bali, yang berisi hari, jam, dan pesan dari pengumuman tersebut, yang dapat diklik dan mengarahkan pengguna ke halaman *announcements*. Pengguna dapat melihat *headline* berita-berita terkait dengan acara WSDC 2017 Bali. Untuk melihat berita tersebut secara penuh, disediakan sebuah tombol yang akan mengarahkan pengguna untuk melihat dan mengunduh berita terkait acara WSDC 2017 Bali. Tangkapan layar halaman *home* dapat dilihat pada Gambar 5.3a. Sebagai perbandingan, tangkapan layar halaman *home* pada aplikasi WSDC 2017 Bali terdahulu dapat dilihat pada Gambar 5.3b.

2. Halaman Announcements

Halaman *announcements* berisi pengumuman-pengumuman terkait dengan acara WSDC 2017 Bali yang disajikan terurut menurun dengan waktu terbaru yang pertama. Tangkapan layar halaman *announcements* dapat dilihat pada Gambar 5.4a. Sebagai perbandingan, tangkapan layar halaman *announcements* pada aplikasi WSDC 2017 Bali terdahulu dapat dilihat pada Gambar 5.4b.

3. Halaman Draw

Halaman *Draw* menampilkan hasil dari pembagian grup oposisi dan proposisi dari negara-negara peserta WSDC 2017 Bali. Tangkapan layar halaman *draw* dapat dilihat pada Gambar 5.5a. Sebagai

perbandingan, tangkapan layar halaman *Draw* pada aplikasi WSDC 2017 Bali terdahulu dapat dilihat pada Gambar 5.5b.

4. Halaman Info

Halaman info menampilkan info-info seperti kontak-kontak penting yang dapat dihubungi, kosa kata dalam Bahasa Indonesia sehari-hari, serta *credits* kepada pembuat aplikasi WSDC 2017 Bali. Tangkapan layar dari halaman info dapat dilihat pada Gambar 5.6a. Sebagai perbandingan, tangkapan layar halaman info pada aplikasi WSDC 2017 Bali terdahulu dapat dilihat pada Gambar 5.6b.

5. Halaman *Result*

Halaman *result* menampilkan hasil dari pertandingan WSDC 2017 Bali pada babak seperdelapan final, seperempat final, dan semifinal. Tangkapan layar dari halaman *result* dapat dilihat pada Gambar 5.7a. Sebagai perbandingan, tangkapan layar halaman *result* pada aplikasi WSDC 2017 Bali terdahulu dapat dilihat pada Gambar 5.7b.

6. Halaman *Schedule*

Halaman *schedule* berisi jadwal acara WSDC 2017 Bali yang ditampilkan berkelompok berdasarkan tanggal dan hari. Jadwal yang ditampilkan berupa waktu mulai dan waktu selesai, lokasi acara, serta nama acara. Pengguna dapat berpindah ke hari manapun untuk melihat jadwal yang ada pada hari tersebut dengan menggulir menyamping pada bagian tanggal dan hari, serta bagian jadwal. Tangkapan layar halaman *schedule* dapat dilihat pada Gambar 5.8a. Lalu sebagai perbandingan, tangkapan layar halaman *schedule* pada aplikasi WSDC 2017 Bali terdahulu dapat dilihat pada Gambar 5.8b.

7. Halaman *Venues*

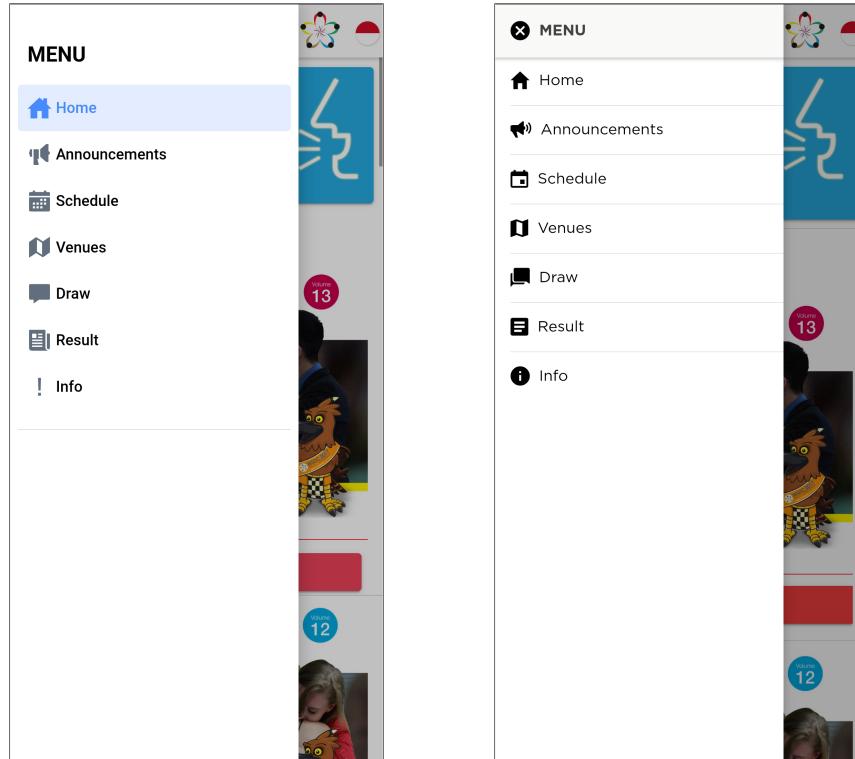
Halaman *venues* berisi kategori *venues* WSDC 2017 Bali. Setiap kategori yang ditampilkan merupakan sebuah tombol yang dapat diklik untuk mengarahkan pengguna ke halaman *venues map*. Tangkapan layar halaman *venues* dapat dilihat pada Gambar 5.9a. Lalu sebagai perbandingan, tangkapan layar halaman *venues* pada aplikasi WSDC 2017 Bali terdahulu dapat dilihat pada Gambar 5.9b.

8. Halaman *Venues Map*

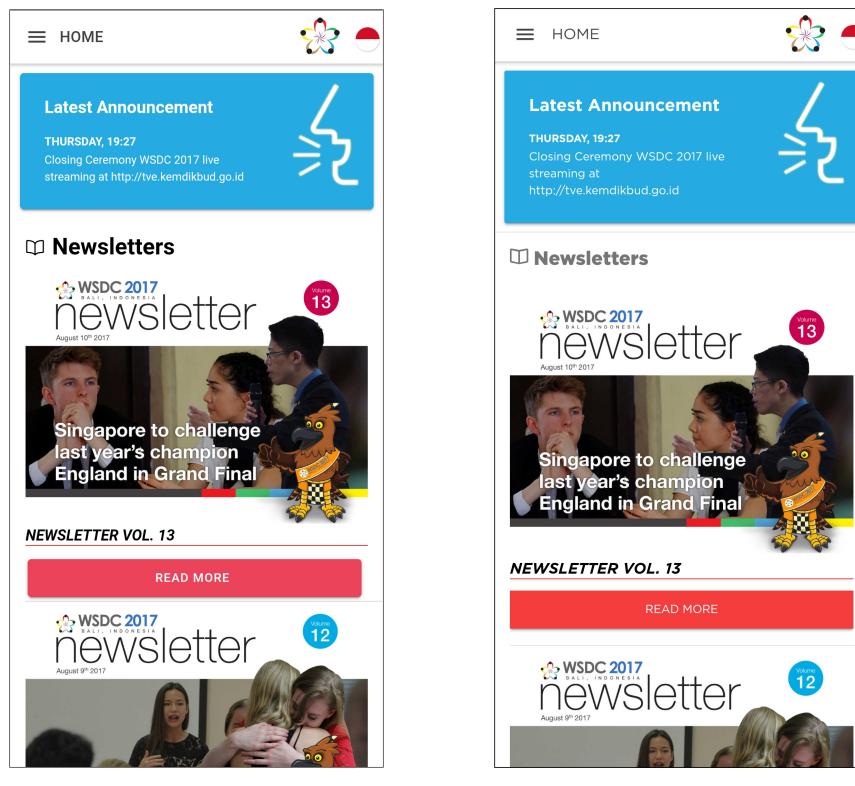
Halaman *venues map* berisi lokasi *venues* yang digunakan oleh WSDC 2017 Bali. Lokasi tersebut ditampilkan dengan peta, dan detail dari lokasi ditampilkan dengan *list* yang berisi nama dan lokasi *venues*, serta jarak dari pengguna ke lokasi *venues*. Tangkapan layar dari halaman *venues map* dapat dilihat pada Gambar 5.10a. Untuk perbandingan, tangkapan layar halaman *venues map* pada aplikasi WSDC 2017 Bali terdahulu dapat dilihat pada Gambar 5.10b.

(a) *Splash Screen Page Terbaru*(b) *Splash Screen Page Terdahulu*

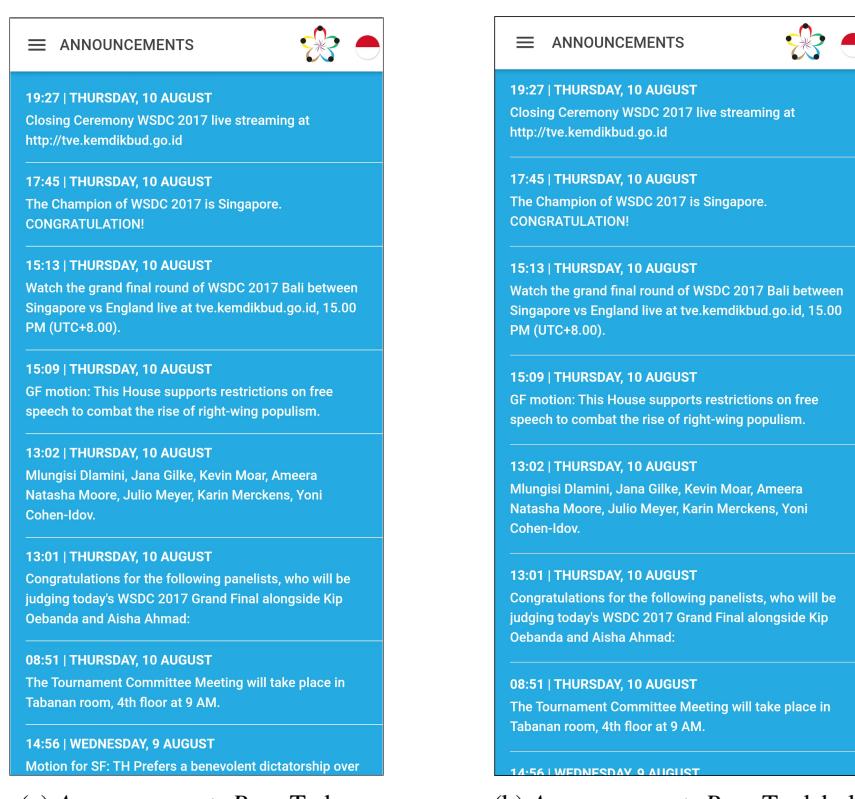
Gambar 5.1: Tangkapan Layar Halaman Splash Screen Aplikasi WSDC 2017 Bali

(a) *Sidemenu Terbaru*(b) *Sidemenu Terdahulu*

Gambar 5.2: Tangkapan Layar Sidemenu Aplikasi WSDC 2017 Bali



Gambar 5.3: Tangkapan Layar Halaman Home Aplikasi WSDC 2017 Bali



Gambar 5.4: Tangkapan Layar Halaman Announcements Aplikasi WSDC 2017 Bali

(a) Draw Page Terbaru

PROPOSITION	OPPOSITION
Argentina	Philippines
Romania	UAE
Bangladesh	Scotland
Wales	Turkey
Ghana	BYE United States
Nepal	Australia
Mexico	Pakistan
Barbados	Sri Lanka
South Korea	Slovenia
Malaysia	Tunisia BYE
Taiwan	Peru
Ireland	Sweden
Bermuda	England
Uganda	Greece

PROPOSITION	OPPOSITION
Czech Republic	China
Mongolia	Denmark
Indonesia	Estonia
Lithuania	Germany
	Hong Kong

(b) Draw Page Terdahulu

PROPOSITION	OPPOSITION
Argentina	Philippines
Romania	UAE
Bangladesh	Scotland
Wales	Turkey
Ghana	BYE United States
Nepal	Australia
Mexico	Pakistan
Barbados	Sri Lanka
South Korea	Slovenia
Malaysia	Tunisia BYE
Taiwan	Peru
Ireland	Sweden
Bermuda	England
Uganda	Greece

Gambar 5.5: Tangkapan Layar Halaman Draw Aplikasi WSDC 2017 Bali

(a) Info Page Terbaru

CO-CONVENORS
wsdc.indonesia@kemdikbud.go.id
Ravio Patra
Kristi Ardiana
Rachmat Nur Cahyo
Nyoman Radjin
Hari Soegiharto
Fonda Ambita Sari

COMMON INDONESIAN PHRASES
I - Saya
You - Kamu
We - Kami
They - Mereka
He / She - Dia
Welcome - Selamat Datang
Hello (general greeting) - Apa kabar?
Hello (on phone) - Halo
How are you? - Apa kabar?
Reply to 'How are you?' - Baik

(b) Info Page Terdahulu

CO-CONVENORS
wsdc.indonesia@kemdikbud.go.id
Ravio Patra
Kristi Ardiana
Rachmat Nur Cahyo
Nyoman Radjin
Hari Soegiharto
Fonda Ambita Sari

COMMON INDONESIAN PHRASES
I - Saya
You - Kamu
We - Kami
They - Mereka
He / She - Dia
Welcome - Selamat Datang
Hello (general greeting) - Apa kabar?
Hello (on phone) - Halo
How are you? - Apa kabar?
Reply to 'How are you?' - Baik
Long time no see - Lama tidak bertemu
What's your name? - Siapa nama anda?
My name is... - Nama saya...
Where are you from? - Anda berasal dari mana?

Gambar 5.6: Tangkapan Layar Halaman Info Aplikasi WSDC 2017 Bali

Semifinals

- USA vs Singapore: 2-5
- South Africa vs England: 2-5

Quarterfinals

- USA vs Peru: 4-1
- South Africa vs Australia: 4-1
- England vs Canada: 3-2
- Singapore vs India: 3-2

Octofinals

Octofinals
 (8 August 2017)

Peru def. Malaysia 4-1
 South Africa def. Denmark 4-1
 Canada def. Philippines 3-2
 India def. China 5-0
 Singapore def. Pakistan 3-2
 England def. Hong Kong 5-0
 Australia def. Greece 4-1
 USA def. Korea 4-1

(a) Result Page Terbaru

Semifinals

- USA vs Singapore: 2-5
- South Africa vs England: 2-5

Quarterfinals

- USA vs Peru: 4-1
- South Africa vs Australia: 4-1
- England vs Canada: 3-2
- Singapore vs India: 3-2

Octofinals

Octofinals
 (8 August 2017)

Peru def. Malaysia 4-1
 South Africa def. Denmark 4-1
 Canada def. Philippines 3-2
 India def. China 5-0
 Singapore def. Pakistan 3-2
 England def. Hong Kong 5-0
 Australia def. Greece 4-1
 USA def. Korea 4-1

(b) Result Page Terdahulu

Gambar 5.7: Tangkapan Layar Halaman *Result* Aplikasi WSDC 2017 Bali

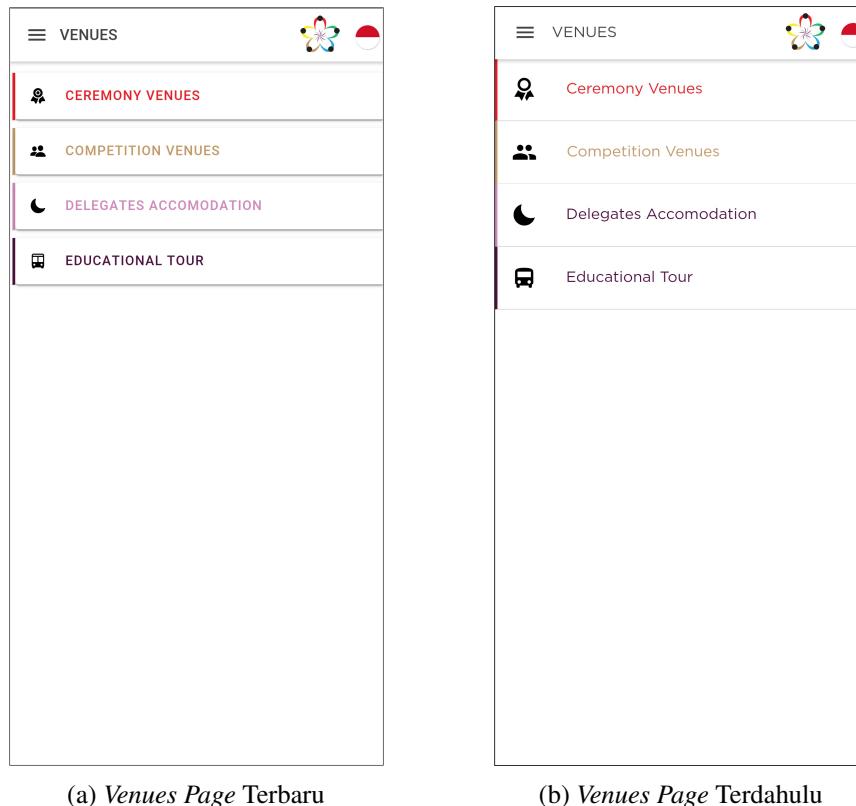
TUE	WED	THU	FRI	SAT	SUN	MON
1	2	3	4	5	6	
00:00 Arrival in Bali 23:59 International Arrival at I Gusti Ngurah Rai Airport						
08:00 Team Registration 22:00 Lobby at Sanur Paradise Plaza Hotel						
14:00 Hotel Check-In 23:59 Lobby at Sanur Paradise Plaza Hotel						
17:30 Dinner 21:00 Griya Agung Ballroom at Sanur Paradise Plaza Hotel						

(a) Schedule Page Terbaru

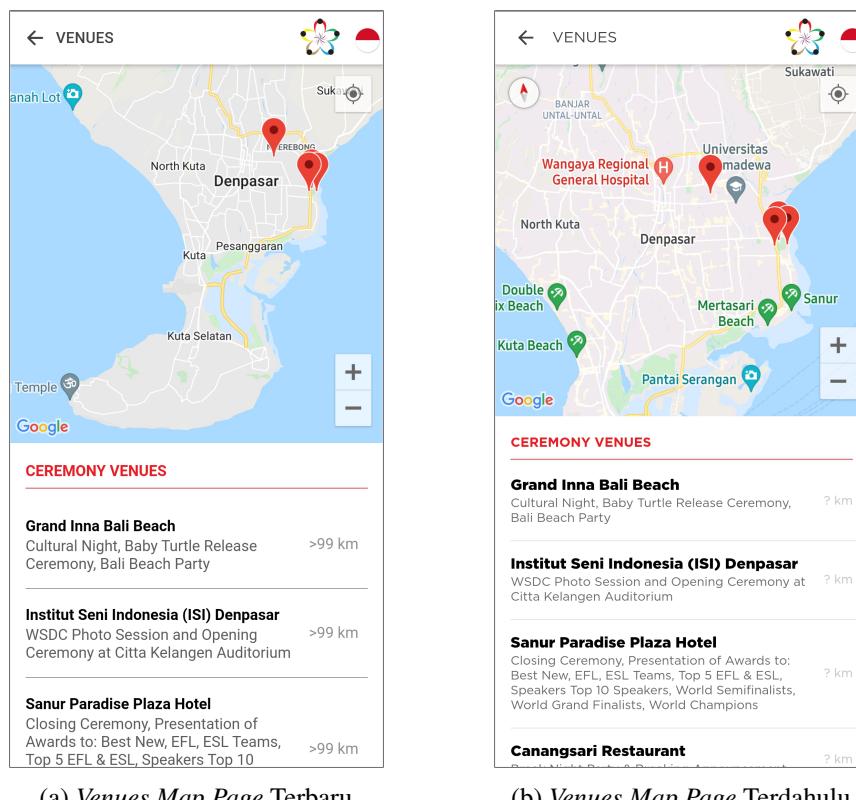
TUE	WED	THU	FRI	SAT	SUN	MON
1	2	3	4	5	6	7
00:00 Arrival in Bali 23:59 International Arrival at I Gusti Ngurah Rai Airport						
08:00 Team Registration 22:00 Lobby at Sanur Paradise Plaza Hotel						
14:00 Hotel Check-In 23:59 Lobby at Sanur Paradise Plaza Hotel						
17:30 Dinner 21:00 Griya Agung Ballroom at Sanur Paradise Plaza Hotel						

(b) Schedule Page Terdahulu

Gambar 5.8: Tangkapan Layar Halaman *Schedule* Aplikasi WSDC 2017 Bali



Gambar 5.9: Tangkapan Layar Halaman *Venues* Aplikasi WSDC 2017 Bali



Gambar 5.10: Tangkapan Layar Halaman *Venues Map* Aplikasi WSDC 2017 Bali

5.2 Pengujian

5.2.1 Pengujian Fungsional

Pengujian fungsional dilakukan untuk mengetahui kesesuaian reaksi perangkat lunak dengan reaksi yang diharapkan berdasarkan aksi pengguna terhadap perangkat lunak. Perangkat yang digunakan untuk melakukan pengujian fungsional ini adalah sebuah perangkat emulator dari Android Studio yaitu Google Pixel 5 dengan versi Android 11, sebuah perangkat emulator Nox Player dengan versi Android 7.1.2, dan sebuah *smartphone* milik penulis yaitu Xiaomi Redmi Note 9 dengan versi Android 11. Pengujian juga dilakukan dengan mode layar lanskap pada ponsel, serta mengecek ukuran teks pada aplikasi ketika pengaturan ukuran teks pada ponsel diperbesar dan diperkecil. Tabel 5.2 merupakan hasil dari 22 tes kasus yang diujikan.

Tabel 5.1: Tabel Pengujian Fungsional

No	Aksi Pengguna	Reaksi yang diharapkan	Reaksi	Perangkat Lunak
1	Pengguna menjalankan aplikasi	Splash Screen ditampilkan dan aplikasi menampilkan halaman home	Sesuai	
2	Pengguna menekan tombol hamburger button di pojok kiri atas aplikasi	Sidemenu terbuka menampilkan menu	Sesuai	
3	Pengguna melakukan swipe dari kiri layar ke kanan layar	Sidemenu terbuka menampilkan menu	Sesuai	
4	Pengguna memilih menu Announcements pada Sidemenu	Aplikasi menampilkan halaman Announcements	Sesuai	
5	Pengguna memilih menu Home pada Sidemenu	Aplikasi menampilkan halaman Home	Sesuai	
6	Pengguna menekan tombol read more pada Home	Aplikasi mengarahkan pengguna untuk melihat newsletter	Sesuai	
7	Pengguna menekan card Latest Announcements	Aplikasi mengarahkan pengguna ke halaman Announcements	Sesuai	
8	Pengguna memilih menu Schedule pada Sidemenu	Aplikasi menampilkan halaman Schedule	Sesuai	
9	Pengguna menekan tombol hari dan tanggal pada halaman Schedule	Aplikasi menampilkan jadwal yang ada pada hari dan tanggal yang dipilih	Sesuai	
10	Pengguna melakukan swipe secara vertical baik dari kiri ke kanan maupun sebaliknya pada halaman Schedule	Aplikasi menampilkan jadwal yang ada pada hari dan tanggal sebelum maupun sesudahnya	Sesuai	
11	Pengguna memilih menu Venues pada Sidemenu	Aplikasi menampilkan halaman Venues	Sesuai	
12	Pengguna memilih kategori venues pada halaman Venues	Aplikasi menampilkan halaman Venues Map yang berisi peta dan lokasi venues	Sesuai	
13	Pengguna menekan tombol lokasi pada map	Aplikasi menampilkan lokasi pengguna pada map dengan titik biru	Sesuai	
14	Pengguna menekan tombol + pada map	Aplikasi melakukan zoom in pada map	Sesuai	

Tabel 5.2: Lanjutan Tabel Pengujian Fungsional dari Halaman Sebelumnya

No	Aksi Pengguna	Reaksi yang diharapkan	Reaksi	Perangkat Lunak
15	Pengguna menekan tombol – pada map	Aplikasi melakukan zoom out pada map	Sesuai	
16	Pengguna menekan nama lokasi venues	Aplikasi melakukan zoom in mengarah ke lokasi yang dituju pada map	Sesuai	
17	Pengguna memilih menu Draw pada Sidemenu	Aplikasi menampilkan halaman Draw	Sesuai	
18	Pengguna memilih menu Result pada Sidemenu	Aplikasi menampilkan halaman Result	Sesuai	
19	Pengguna memilih menu Info pada Sidemenu	Aplikasi menampilkan halaman Info	Sesuai	
20	Pengguna menekan nomor telepon pada halama info	Aplikasi mengarahkan pengguna ke aplikasi pemanggilan	Sesuai	
21	Ukuran teks diperbesar pada pengaturan ponsel	Ukuran teks pada aplikasi membesar sesuai dengan pengaturan pada ponsel	Sesuai	
22	Ukuran teks diperkecil pada pengaturan ponsel	Ukuran teks pada aplikasi mengecil sesuai dengan pengaturan pada ponsel	Sesuai	
23	Mencoba mode lanskap	Aplikasi ditampilkan dalam mode lanskap	Sesuai	

5.2.2 Pengujian Eksperimental

Pengujian eksperimental dilakukan terhadap pengguna *smartphone* dengan sistem operasi Android. Metode pengujian dilakukan dengan cara menyebarluaskan aplikasi yang dapat diunduh melalui Google Drive¹. Responden yang dipilih merupakan sembilan orang mahasiswa dari berbagai jurusan di beberapa universitas di Indonesia dengan rentang usia 21 sampai 23 tahun. Kemudian, pengguna diminta untuk mengunduh dan menjalankan aplikasi tersebut. Pengguna juga diminta untuk mengunduh aplikasi WSDC 2017 Bali terdahulu melalui Google Play Store² dan menjalankannya. Setelah itu pengguna diminta untuk membandingkan kedua aplikasi tersebut, dan mengisi beberapa pertanyaan terkait pengalaman menggunakan aplikasi WSDC 2017 Bali melalui Google Form. Berikut ini merupakan pertanyaan dan rangkuman jawaban dari hasil pengujian eksperimental terhadap sembilan responden sebagai berikut:

1. Apa versi Android smartphone Anda?

Seorang responden menjawab versi Android 5.1, seorang menjawab versi Android 8.0, seorang menjawab versi Android 8.1, seorang menjawab versi Android 10, dan empat orang menjawab versi Android 11.

2. Saat pertama kali membuka aplikasi, apakah aplikasi WSDC 2017 Bali terbaru menampilkan logo WSDC?

Semua responden menjawab aplikasi WSDC 2017 Bali terbaru menampilkan logo WSDC.

3. Apakah semua halaman memiliki isi nya masing-masing, dan tidak ada halaman yang isinya kosong?

Semua responden menjawab tidak ada halaman yang tidak memiliki isi.

4. Apakah tombol GPS, zoom in, zoom out, dan lokasi venues yang terdapat pada menu Venues dapat berfungsi dengan baik?

Semua responden menjawab semua tombol berfungsi dengan normal.

¹Tautan Google Drive aplikasi WSDC 2017 Bali dengan Ionic 6 yang diujikan kepada responden: <https://drive.google.com/file/d/1Np29U2dg58Pryp1cbrs-M3XfUcm39cSZ/view?usp=sharing>

²Tautan Google Play Store aplikasi WSDC 2017 Bali terdahulu: <https://play.google.com/store/apps/details?id=org.wsdc2017indonesia.app>

5. Apakah Anda mengalami *crash*, *forced close*, atau kendala lain saat menggunakan aplikasi WSDC 2017 Bali terbaru?

Sebanyak delapan responden menjawab tidak ada crash, forced close, atau kendala lain saat menggunakan aplikasi WSDC 2017 Bali terbaru, dan ada satu responden yang menjawab iya, namun tidak menjelaskan kendala apa yang terjadi.

6. Apakah ada perbedaan positif yang signifikan dibandingkan dengan aplikasi terdahulu?

Dua orang responden berpendapat bahwa tampilan *sidemenu* tampak lebih segar dan menarik. Lalu sebanyak satu responden berpendapat bahwa tampilan *icon* terlihat lebih beragam dan menarik. Kemudian sebanyak empat responden berpendapat bahwa aplikasi WSDC 2017 Bali terbaru dapat dibuka dengan lebih cepat dibandingkan dengan aplikasi terdahulu. Lalu sebanyak satu responden berpendapat bahwa perubahan aplikasi menjadi lebih baik dari sebelumnya, satu responden menjawab perubahan yang terjadi hanya sedikit, dan satu responden menjawab tidak ada perubahan positif yang dirasakan.

7. Apakah ada perbedaan negatif yang signifikan dibandingkan dengan aplikasi terdahulu?

Sebanyak empat orang responden menjawab bahwa tidak ada perubahan negatif pada aplikasi WSDC 2017 Bali terbaru. Seorang responden menjawab *font* tulisan lebih kaku, dan halaman *draw* kualitasnya terlihat lebih rendah dibandingkan aplikasi terdahulu. Lalu seorang responden menjawab tampilan pada aplikasi yang terbaru dari menu yang ada di masing-masing *Venues* sedikit aneh, karena nama tempat dan alamatnya saling berdekatan tanpa ada jarak spasi. Dan seorang responden menjawab pemakaian aplikasi terbaru lebih boros.

8. Secara keseluruhan, dengan skala 1-5, seberapa baik aplikasi WSDC 2017 Bali terbaru dibandingkan dengan aplikasi terdahulu?

Seorang responden menjawab netral dengan skala 3, enam orang responden menjawab dengan skala 4, dan dua orang responden menjawab aplikasi WSDC 2017 Bali lebih baik dibandingkan aplikasi terdahulu dengan skala 5.

9. Apakah Anda lebih memilih menggunakan aplikasi WSDC 2017 Bali terdahulu, atau yang terbaru?

Sebanyak delapan responden memilih untuk menggunakan aplikasi WSDC 2017 Bali terbaru, sedangkan satu responden memilih untuk menggunakan aplikasi WSDC 2017 Bali terdahulu.

10. Apakah terdapat kritik dan saran terhadap aplikasi WSDC 2017 Bali terbaru?

Terdapat beberapa kritik dan saran dari responden sebagai berikut:

- (a) Dapat memilih font yang lebih menarik dan nyaman untuk dibaca.
- (b) Pada halaman *Result* diharapkan agar memiliki tampilan lebih menarik lagi.
- (c) Ukuran aplikasi bisa dikecilkan.
- (d) Pada halaman *venues* dimana tulisan headline dari list map dijauhkan sedikit dari tulisan body karena terlalu dekat.
- (e) Ukuran teks bisa tetap sama ukurannya untuk setiap kondisi ukuran teks pada pengaturan ponsel.
- (f) Membuat aplikasi untuk acara yang masih atau akan berlangsung, karena informasi pada aplikasi WSDC 2017 Bali sudah tidak *update*.

Namun menurut pengamatan penulis terhadap kritik dan saran pada poin c, bahwa ukuran aplikasi WSDC 2017 Bali terbaru sudah cukup kecil, yaitu 25MB. Sedangkan untuk poin a dan b bersifat subjektif, sehingga tidak akan diimplementasikan oleh penulis.

BAB 6

KESIMPULAN DAN SARAN

6.1 Kesimpulan

Dari hasil pembangunan aplikasi WSDC 2017 Bali menggunakan Ionic 6, didapatkan kesimpulan sebagai berikut:

1. Telah berhasil melakukan pembaruan aplikasi WSDC 2017 Bali dengan Ionic Framework versi 6 yang sebelumnya menggunakan Ionic Framework versi 3.
2. Aplikasi WSDC 2017 Bali telah dapat dijalankan pada perangkat dengan sistem operasi Android.

6.2 Saran

Dari hasil penelitian dan pengujian termasuk dengan pengujian terhadap responden, berikut ini merupakan beberapa saran untuk pengembangan lebih lanjut:

1. Pada halaman *venues* dimana tulisan headline dari list map dijauhkan sedikit dari tulisan body karena terlalu dekat.
2. Dapat mengecilkan ukuran teks yang terlalu besar ketika pengaturan teks di ponsel diperbesar.
3. Karena acara WSDC 2017 Bali sudah selesai, dapat dibuat untuk acara lainnya yang masih berjalan.

DAFTAR REFERENSI

- [1] Emmit A. Scott, J. (2015) *SPA Design and Architecture: Understanding single-page web applications*, 1st edition. Manning Publications, New York, USA.
- [2] Wargo, J. M. (2014) *Apache Cordova API Cookbook*, 1st edition. Pearson Education, Inc., New Jersey, USA.
- [3] World Schools Debate Championship (2021) WSDC. <https://wsdcdebate.org/history>. [Online; diakses 8-Juli-2021].
- [4] Waranashiwar, J. dan Ukey, M. (2018) Ionic framework with angular for hybrid app development. *International Journal of New Technology and Research*, **4**, 01–02.
- [5] Yusuf, S. (2016) *Ionic Framework By Example*, 1st edition. Pact Publishing Ltd., Birmingham, UK.
- [6] Griffith, C. (2017) *Mobile App Development with Ionic : Cross-Platform Apps with Ionic, Angular and Cordova*, 1st edition. O'Reilly Media, Inc., California, USA.
- [7] Grønli, T.-M., Biørn-Hansen, A., dan Majchrzak, T. A. (2019) Median trajectories using well-visited regions and shortest paths software development for mobile computing the internet of things and wearable devices: Inspecting the past to understand the future. *Proceedings of the 52nd Hawaii International Conference on System Sciences*, Grand Wailea, Hawaii, 8–11 January, pp. 7451–7460. University of Hawaii, Manoa.
- [8] Wohlgethan, E. (2018) Supporting web development decisions by comparing three major javascript frameworks: Angular, react and vue.js. Thesis. Hochschule für angewandte Wissenschaften Hamburg, Germany.
- [9] Moiseev, A. dan Fain, Y. (2018) *Angular Development with TypeScript*, 2nd edition. Manning Publications, New York, USA.
- [10] Prusty, N. (2015) *Learning ECMAScript 6*, 1st edition. Pact Publishing Ltd., Birmingham, UK.
- [11] Kunz, G. (2018) *Mastering Angular Components: Build component-based user interfaces using Angular*, 2nd edition. Pact Publishing Ltd., Birmingham, UK.
- [12] Savkin, V. (2017) *Angular Router*, 1st edition. Pact Publishing Ltd., Birmingham, UK.
- [13] Huber, S., Demetz, L., dan Felderer, M. (2021) Pwa vs the others: A comparative study on the ui energy-efficiency of progressive web apps. *Web Engineering*, Switzerland, 11 May, pp. 464–479. Springer International Publishing.
- [14] Gonsalves, M. (2018) Evaluating the mobile development frameworks apache cordova and flutter and their impact on the development process and application characteristics. Thesis. California State University, Chico, California, USA.

LAMPIRAN A

KODE PROGRAM

A.1 Komponen Announcements

```
1 import { HttpClient } from '@angular/common/http';
2 import { Component, OnInit } from '@angular/core';
3 import { Storage } from '@ionic/storage';
4 import { ToastController } from '@ionic/angular';
5
6 @Component({
7   selector: 'app-announcement',
8   templateUrl: './announcement.page.html',
9   styleUrls: ['./announcement.page.scss'],
10 })
11
12 export class AnnouncementPage implements OnInit {
13   announcements: Array<{ localtime: string, message: string }>;
14
15   constructor(private http: HttpClient, private storage: Storage, public toastController
16     : ToastController) { }
17
18   ngOnInit(): void {
19     this.storage.get('wsdcDataStorage').then((data) => {
20       this.announcements = data.announcements;
21     })
22   }
23
24   doRefresh(refresher) {
25     console.log('Begin async operation');
26     this.http.get('https://wsdc.dnartworks.com/wsdc_data.json').subscribe(
27       (data: any) => {
28         this.storage.clear();
29         this.storage.set('wsdcDataStorage', data);
30         this.announcements = data.announcements;
31         console.log("Data updated!");
32         if (refresher != 0)
33           refresher.target.complete();
34       },
35       (error) => {
36         this.presentConnectionAlert();
37         refresher.target.complete();
38         console.log('error in XMLHttpRequest JSON');
39       });
40   setTimeout(() => {
41     console.log('Async operation has ended');
42     refresher.target.complete();
43   }, 30000);
44 }
45
46 async presentConnectionAlert() {
47   let toast = await this.toastController.create({
```

```

47     message: 'Failed to refresh information',
48     duration: 3000
49   });
50   toast.present();
51 }
52
53 formatDatetime(sqlDatetime: string) {
54   var date = new Date(sqlDatetime.substring(0, 10));
55   var dayNames = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
56   "Saturday"];
57   var monthNames = ["January", "February", "March", "April", "May", "June", "July",
58   "August", "September", "October", "November", "December"];
59   var dayOfWeek = date.getDay();
60   var day = date.getDate();
61   var monthIndex = date.getMonth();
62   var time=sqlDatetime.substring(16,4)
63   return time.substring(7) + ' | ' + dayNames[dayOfWeek] + ', ' + day + ' ' +
monthNames[monthIndex];
}
}

```

Kode A.1: announcement.page.ts

```

1 <ion-header>
2   <ion-toolbar>
3     <ion-buttons slot="start">
4       <ion-menu-button></ion-menu-button>
5     </ion-buttons>
6     <ion-title>Announcements</ion-title>
7   </ion-toolbar>
8 </ion-header>
9
10 <ion-content>
11   <ion-refresher slot="fixed" (ionRefresh)="doRefresh($event)">
12     <ion-refresher-content></ion-refresher-content>
13   </ion-refresher>
14   <ion-list>
15     <ion-item color="wsdc-blue" *ngFor="let announcement of announcements; let i =
index">
16       <ion-label class="ion-text-wrap">
17         <h3>{{ formatDatetime(announcement.localtime) }}</h3>
18         <p>{{ announcement.message }}</p>
19       </ion-label>
20     </ion-item>
21   </ion-list>
22 </ion-content>

```

Kode A.2: announcement.page.html

A.2 Komponen Draw

```

1 import { Component, ElementRef, OnInit, ViewChild } from '@angular/core';
2 import { LoadingController } from '@ionic/angular';
3 import { Storage } from '@ionic/storage';
4
5 @Component({
6   selector: 'app-draw',
7   templateUrl: './draw.page.html',
8   styleUrls: ['./draw.page.scss'],
9 })
10

```

```

11 export class DrawPage implements OnInit {
12   @ViewChild('drawIFrame') drawIFrame: ElementRef;
13   constructor(private storage: Storage, public loadingController: LoadingController) {
14     }
15   ngOnInit() {
16     this.storage.get('wsdcDataStorage').then((data) => {
17       this.drawIFrame.nativeElement.contentWindow.location.assign(data.draws);
18     });
19     this.presentLoading();
20   }
21
22   async presentLoading() {
23     const loading = await this.loadingController.create({
24       message: 'Please wait...',
25       backdropDismiss: true // If true, the loading indicator will be dismissed when
26       the backdrop is clicked.
27     );
28     await loading.present();
29     setTimeout(() => {
30       loading.dismiss();
31     }, 1000);
32   }
33
34   onDrawIframeLoad() {
35     let doc = this.drawIFrame.nativeElement.contentWindow.document;
36     let elements = [
37       doc.getElementById('header'),
38       doc.getElementById('page_header'),
39       doc.getElementById('footer')
40     ];
41     elements.forEach(function (element) {
42       if (element) {
43         element.style.display = 'none';
44       }
45     })
46     this.drawIFrame.nativeElement.style.display = 'block';
47   }
48 }s

```

Kode A.3: draw.page.ts

```

1 <ion-header>
2   <ion-toolbar>
3     <ion-buttons slot="start">
4       <ion-menu-button></ion-menu-button>
5     </ion-buttons>
6     <ion-title>Draw</ion-title>
7   </ion-toolbar>
8 </ion-header>
9
10 <ion-content>
11   <iframe #drawIFrame (load)="onDrawIframeLoad()"></iframe>
12 </ion-content>

```

Kode A.4: draw.page.html

A.3 Komponen Home

```

1 import { Component, OnInit } from '@angular/core';
2 import { HttpClient } from '@angular/common/http';
3 import { Browser } from '@capacitor/browser';
4 import { Storage } from '@ionic/storage';
5 import { Router } from '@angular/router';
6 import { SplashScreen } from '@capacitor/splash-screen';
7 import { ToastController } from '@ionic/angular';
8
9 @Component({
10   selector: 'app-home',
11   templateUrl: './home.page.html',
12   styleUrls: ['./home.page.scss'],
13 })
14
15 export class HomePage implements OnInit {
16   wsdcData:any;
17
18   constructor(private http: HttpClient,private storage: Storage,private router: Router
19   ,public toastController: ToastController) { }
20
21   ionViewDidEnter(){
22     SplashScreen.hide()
23   }
24
25   ngOnInit(){
26     this.storage.get('wsdcDataStorage').then((data) => {
27
28       if(data == null){
29         //Default from asset
30         this.http.get('../assets/json/wsdc_data.json').subscribe((data: any) => {
31           this.wsdcData = data;
32           this.storage.set('wsdcDataStorage',data);
33         },
34         error => {
35           this.showToast('Failed to refresh information from local storage');
36         });
37       }else{
38         this.wsdcData = data;
39       }
40
41       // Refresh data
42       setTimeout(() => {
43         this.http.get('http://wsdc.dnartworks.com/wsdc_data.json')
44         .subscribe((data: any) => {
45           this.storage.set('wsdcDataStorage', data);
46           this.wsdcData = data;
47         },
48         error => {
49           this.showToast('Failed to refresh information');
50         });
51       }, 1000);
52     })
53   }
54
55   doRefresh(refresher) {
56     this.http.get('https://wsdc.dnartworks.com/wsdc_data.json').subscribe((data: any)
57     => {
58       this.storage.clear();
59       this.storage.set('wsdcDataStorage',data);
      this.wsdcData = data;
    })
  }

```

```

60     },
61     error => {
62       // Timeout or no connection
63       this.showToast('Failed to refresh information');
64       refresher.complete();
65     });
66
67     setTimeout(() => {
68       refresher.target.complete();
69     }, 2000);
70   }
71
72   async showToast(message: string, duration: number=3000) {
73     let toast = await this.toastController.create({
74       message: message,
75       duration: duration
76     });
77     toast.present();
78   }
79
80   formatDatetime(sqlDatetime: string) {
81     if (sqlDatetime === null) {
82       return null;
83     }
84     var time=sqlDatetime.substring(16,4)
85     var date = new Date(sqlDatetime.substring(0, 10));
86     var dayNames = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
87     "Saturday"];
88     var dayOfWeek = date.getDay();
89     return dayNames[dayOfWeek] + ', ' + time.substring(7);
90   }
91
92   // ionic 6 : use capacitor in app browser
93   launch(newsUrl: string) {
94     Browser.open({ url: newsUrl });
95   }
96
97   // ionic 6 : use ionic angular router for change page
98   onAnnouncementClick() {
99     this.router.navigate(['announcement']);
10   }
100 }

```

Kode A.5: home.page.ts

```

1 <ion-header>
2   <ion-toolbar>
3     <ion-buttons slot="start">
4       <ion-menu-button></ion-menu-button>
5     </ion-buttons>
6     <ion-title>Home</ion-title>
7   </ion-toolbar>
8 </ion-header>
9
10 <ion-content>
11   <ion-refresher slot="fixed" (ionRefresh)="doRefresh($event)">
12     <ion-refresher-content></ion-refresher-content>
13   </ion-refresher>
14   <ion-card (click)="onAnnouncementClick()">
15     <ion-grid>
16       <ion-row>
17         <ion-col size="9">
18           <ion-card-header>

```

```

19         <ion-card>title>Latest Announcement</ion-card>titlecontent>
23         <h3>{{ formatDatetime(wsdcdData?.announcements[0].localtime) }}</h3>
24         <p>{{ wsdcdData?.announcements[0].message }}</p>
25     </ion-card>content>
26   </ion-col>
27   <ion-col size="3">
28     
29   </ion-col>
30 </ion-row>
31 </ion-grid>
32 </ion-card>
33
34 <ion-list>
35   <ion-list-header>
36     <ion-icon name="book-outline"></ion-icon>
37     <ion-label class="label_newsletters"><h1><b> Newsletters</b></h1></ion-label>
38   </ion-list-header>
39   <ion-item *ngFor="let wsdcNews of wsdcdData?.newsletters;">
40     <div class="newsletters" >
41       
42         <h2 text-wrap>{{wsdcNews.title}}</h2> <br>
43         <ion-button class="ion-padding-end" full block color="danger" (click)="launch(wsdcNews.url)">Read More</ion-button>
44     </div>
45   </ion-item>
46 </ion-list>
47 </ion-content>

```

Kode A.6: home.page.html

A.4 Komponen Info

```

1 import { Component, ViewEncapsulation } from '@angular/core';
2 import { Storage } from '@ionic/storage';
3
4 @Component({
5   selector: 'app-info',
6   templateUrl: './info.page.html',
7   styleUrls: ['./info.page.scss'],
8   encapsulation: ViewEncapsulation.None,
9 })
10
11 export class InfoPage{
12   wsdcdInfoData: any;
13
14   constructor(private storage: Storage) {
15
16     this.storage.get('wsdcDataStorage').then((data) => {
17       this.wsdcdInfoData = data.info;
18       this.wsdcdInfoData = this.wsdcdInfoData.replace(new RegExp('icon-telephone', 'g'),
19         '');
20     })
21   }
22 }

```

Kode A.7: info.page.ts

```

1 <ion-header>
2   <ion-toolbar>
3     <ion-buttons slot="start">
4       <ion-menu-button></ion-menu-button>
5     </ion-buttons>
6     <ion-title>Info</ion-title>
7   </ion-toolbar>
8 </ion-header>
9
10 <ion-content>
11   <ion-grid>
12     <ion-row>
13       <div [innerHTML]="wsdcInfoData"></div>
14     </ion-row>
15   </ion-grid>
16 </ion-content>

```

Kode A.8: info.page.html

A.5 Komponen Result

```

1 import { Component, ElementRef, OnInit, ViewChild } from '@angular/core';
2 import { Storage } from '@ionic/storage';
3 import { LoadingController } from '@ionic/angular';
4
5 @Component({
6   selector: 'app-result',
7   templateUrl: './result.page.html',
8   styleUrls: ['./result.page.scss'],
9 })
10
11 export class ResultPage implements OnInit {
12   @ViewChild('resultIFrame') resultIFrame: ElementRef;
13
14   constructor(private storage: Storage, public loadingController: LoadingController) {
15     }
16
17   ngOnInit() {
18     this.storage.get('wsdcDataStorage').then((data) => {
19       this.resultIFrame.nativeElement.contentWindow.location.assign(data.results);
20     });
21     this.presentLoading();
22   }
23
24   async presentLoading() {
25     const loading = await this.loadingController.create({
26       message: 'Please wait...',
27       backdropDismiss: true
28     });
29
30     await loading.present();
31
32     setTimeout(() => {
33       loading.dismiss();
34     }, 500);
35   }
36
37   onResultIframeLoad() {
38     let doc = this.resultIFrame.nativeElement.contentWindow.document;
39     let elements = [

```

```

39     doc.getElementById('header'),
40     doc.getElementById('page_header'),
41     doc.getElementById('footer')
42   ];
43   elements.forEach(function (element) {
44     if (element) {
45       element.style.display = 'none';
46     }
47   })
48   this.resultIFrame.nativeElement.style.display = 'block';
49 }
50
51 }

```

Kode A.9: result.page.ts

```

1 <ion-header>
2   <ion-toolbar>
3     <ion-buttons slot="start">
4       <ion-menu-button></ion-menu-button>
5     </ion-buttons>
6     <ion-title>Result</ion-title>
7   </ion-toolbar>
8 </ion-header>
9
10 <ion-content>
11   <iframe #resultIFrame (load)="onResultIframeLoad()"></iframe>
12 </ion-content>

```

Kode A.10: result.page.html

A.6 Komponen Schedule

```

1 import { Component, ElementRef } from '@angular/core';
2 import { ViewChild } from '@angular/core';
3 import { IonSlides } from '@ionic/angular';
4 import { Storage } from '@ionic/storage';
5
6 @Component({
7   selector: 'app-schedule',
8   templateUrl: './schedule.page.html',
9   styleUrls: ['./schedule.page.scss'],
10 })
11 export class SchedulePage implements OnInit{
12   @ViewChild('scheduleSlider', { static: false }) slider: IonSlides;
13   @ViewChild('segmentContainer', { static: false }) segmentContainer: ElementRef;
14   schedules: any;
15   slideOpts = {
16     initialSlide: 0,
17     speed: 400,
18   };
19   selectedSegmentIdx: any;
20   currentIndex: any;
21
22   constructor(private storage: Storage) {
23
24   }
25
26   ngOnInit() {
27     this.storage.get('wsdcDataStorage').then((val) => {
28       this.schedules = val.schedules;

```

```

29     this.selectedSegmentIdx = 0;
30   });
31 }
32
33 getDayName(sqlDate: string) {
34   var date = new Date(sqlDate);
35   var dayNames = ["Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"];
36   var dayOfWeek = date.getDay();
37   return dayNames[dayOfWeek];
38 }
39
40 getDate(sqlDate: string) {
41   var date = new Date(sqlDate);
42   return date.getDate();
43 }
44
45 onSlideChanged() {
46   this.slider.getActiveIndex().then((index: number) => {
47     this.currentIndex = index;
48     document.getElementById(this.currentIndex).scrollIntoView({
49       behavior: 'smooth',
50       block: 'center',
51       inline: 'center'
52     });
53     this.selectedSegmentIdx = index;
54   });
55 }
56
57 onSegmentChanged(segmentButton) {
58   this.slider.slideTo(segmentButton.detail.value);
59 }
60 }

```

Kode A.11: schedule.page.ts

```

1 <ion-header>
2   <ion-toolbar>
3     <ion-buttons slot="start">
4       <ion-menu-button></ion-menu-button>
5     </ion-buttons>
6     <ion-title>Schedule</ion-title>
7   </ion-toolbar>
8 </ion-header>
9
10 <ion-content>
11   <div id="schedulesContainer">
12     <div id="schedulesSegments" slot="fixed">
13       <ion-segment #segmentContainer scrollable [(ngModel)]="selectedSegmentIdx" (
14         ionChange)="onSegmentChanged($event)">
15         <ion-segment-button *ngFor="let schedule of schedules; let i = index;" [value
16           ]="i" [id]="i">
17           <ion-label>
18             <div class="day">{{ getDayName(schedule.date) }}</div>
19             <div class="date">{{ getDate(schedule.date) }}</div>
20           </ion-label>
21         </ion-segment-button>
22       </ion-segment>
23     </div>
24     <div id="schedulesSlides">
25       <ion-slides #scheduleSlider [options]="slideOpts" (ionSlideDidChange)="
onSlideChanged()">
26         <ion-slide *ngFor="let schedule of schedules;">
27           <ion-list>

```

```

26         <ion-item *ngFor="let agenda of schedule.agenda;" >
27             <ion-note item-start>
28                 {{agenda.start}} <br>
29                 {{agenda.end}}
30             </ion-note>
31             <ion-label class="ion-text-wrap">
32                 <h3>{{agenda.title}}</h3>
33                 <p>{{agenda.subtitle}}</p>
34             </ion-label>
35         </ion-item>
36     </ion-list>
37 </ion-slide>
38 </ion-slides>
39 </div>
40 </div>
41 </ion-content>
```

Kode A.12: schedule.page.html

A.7 Komponen Venues

```

1 import { Component, OnInit } from '@angular/core';
2 import { Router, NavigationExtras } from '@angular/router';
3 import { Storage } from '@ionic/storage';
4
5 @Component({
6   selector: 'app-venues',
7   templateUrl: './venues.page.html',
8   styleUrls: ['./venues.page.scss'],
9 })
10 export class VenuesPage implements OnInit{
11   venuesData: Array<{ id: string, name: string, icon: string, geojson: any, colorIdx: number }>;
12   valVenues: any;
13   constructor(private router: Router,private storage: Storage) {
14
15 }
16
17 ngOnInit() {
18   this.storage.get('wsdcDataStorage').then((val) => {
19     this.valVenues = val.venues;
20     this.venuesData = [];
21     let currColorIdx: number = 1;
22     for (let venue of this.valVenues) {
23       this.venuesData.push({
24         id: venue.id,
25         name: venue.name,
26         icon: venue.icon,
27         geojson: venue.geojson,
28         colorIdx: currColorIdx,
29       });
30       if (currColorIdx > 4) {
31         currColorIdx = 1;
32       } else {
33         currColorIdx++;
34       }
35     }
36   })
37 }
38
39   itemTapped(wsdcVenue) {
```

```

40 // this.router.navigate(['venues-map', id])
41 let navigationExtras: NavigationExtras = {
42   state: {
43     venuesData: wsdcVenue
44   }
45 };
46 this.router.navigate(['venues-map'], navigationExtras);
47 }
48 }
```

Kode A.13: venues.page.ts

```

1 <ion-header>
2   <ion-toolbar>
3     <ion-buttons slot="start">
4       <ion-menu-button></ion-menu-button>
5     </ion-buttons>
6     <ion-title>Venues</ion-title>
7   </ion-toolbar>
8 </ion-header>
9
10 <ion-content>
11   <ion-grid>
12     <ion-row>
13       <ion-list no-lines>
14         <ion-button color="white" id="{{wsdcVenue.id}}" *ngFor="let wsdcVenue of
15           venuesData" (click)="itemTapped(wsdcVenue)">
16           <ion-icon name="{{wsdcVenue.icon}}></ion-icon>
17           <span>{{ wsdcVenue.name }}</span>
18         </ion-button>
19       </ion-list>
20     </ion-row>
21   </ion-grid>
22 </ion-content>
```

Kode A.14: venues.page.html

A.8 Komponen Venues Map

```

1 import { Component, OnInit } from '@angular/core';
2 import { ActivatedRoute, Router } from '@angular/router';
3 import { Storage } from '@ionic/storage';
4 import { CapacitorGoogleMaps } from "@capacitor-community/google-maps";
5 import { Geolocation } from '@capacitor/geolocation';
6
7 @Component({
8   selector: 'app-venues-map',
9   templateUrl: './venues-map.page.html',
10  styleUrls: ['./venues-map.page.scss'],
11 })
12 export class VenuesMapPage implements OnInit{
13   venuesPage: any;
14   venuesMapsDetail: any;
15   userCoordinatesLat: any;
16   userCoordinatesLng: any;
17   mapid: any;
18   userDistanceTo: string[];
19   itemCounter: number;
20   items: Array<{id: string, idcolor:number}>;
21   pageTitleColors: Array<string> = ["#ec1c24", "#c49a6c", "#d189bb", "#4d113f"];
22 }
```

```
23 constructor(private activatedRoute: ActivatedRoute, private router: Router,private
24 storage: Storage) {
25
26     //Get data from venues page.
27     this.items = [];
28     this.activatedRoute.queryParams.subscribe(params => {
29         if (this.router.getCurrentNavigation().extras.state) {
30             this.venuesPage = this.router.getCurrentNavigation().extras.state.venuesData;
31         }
32
33         this.items.push({
34             id: this.venuesPage.id,
35             idcolor: this.venuesPage.colorIdx
36         });
37
38         document.getElementById("pagetitle").style.color = this.pageTitleColors[this.
39 items[0].idcolor-1];
40     });
41
42     //save user lat & lng location
43     const printCurrentPosition = async () => {
44         const coordinates = await Geolocation.getCurrentPosition();
45         this.userCoordinatesLat = coordinates.coords.latitude;
46         this.userCoordinatesLng = coordinates.coords.longitude;
47     };
48
49     printCurrentPosition();
50     Geolocation.requestPermissions();
51 }
52
53 ngOnInit() {
54     // Select data from storage
55     this.storage.get('wsdcDataStorage').then((data) => {
56         this.venuesMapsDetail = data.venues;
57         this.venuesMapsDetail = this.venuesMapsDetail.filter(d=> d.id==this.items[0].id);
58     });
59 }
60
61 ionViewDidEnter() {
62     // create map and add marker
63     const initializeMap = async () => {
64         this.itemCounter = 0;
65         await CapacitorGoogleMaps.initialize({
66             key: "YOUR_IOS_MAPS_API_KEY",
67             devicePixelRatio: window.devicePixelRatio,
68         });
69         const element = document.getElementById("mapContainer");
70         const boundingRect = element.getBoundingClientRect();
71         try {
72             const result = await CapacitorGoogleMaps.createMap({
73                 boundingRect: {
74                     width: Math.round(boundingRect.width),
75                     height: Math.round(boundingRect.height),
76                     x: Math.round(boundingRect.x),
77                     y: Math.round(boundingRect.y),
78                 },
79                 cameraPosition:{
80                     target:{ //Kuta, Bali -8.722396, 115.17671
81                         latitude:-8.722396,
82                         longitude: 115.17671
83                     },
84                     zoom:11
85                 }
86             });
87         }
88     };
89 }
```

```
83     },
84     preferences:{
85       controls:{
86         isCompassButtonEnabled:true,
87         isMyLocationButtonEnabled:true,
88         isZoomButtonsEnabled:true
89       },
90       appearance:{
91         isMyLocationDotShown:true
92       }
93     },
94   });
95
96   element.style.background = "";
97   element.setAttribute("data-maps-id", result.googleMap.mapId);
98   this.mapid = result.googleMap.mapId;
99   console.log(this.venuesMapsDetail);
100
101 //add marker to each location
102 for(let venue of this.venuesMapsDetail){
103   for(let venuesMarker of venue.geojson.features){
104     this.itemCounter +=1;
105     const koor = venuesMarker.geometry.coordinates;
106     CapacitorGoogleMaps.addMarker({
107       mapId:result.googleMap.mapId,
108       position:{
109         latitude: koor[1],
110         longitude: koor[0],
111       },
112       preferences:{
113         title: venuesMarker.properties.Name
114       },
115     });
116   }
117 }
118 } catch (e) {
119   alert("Map failed to load");
120 }
121
122 // Insert distance to each location
123 this.userDistanceTo = new Array(this.itemCounter);
124 let counter = 0;
125 for(let venue of this.venuesMapsDetail){
126   for(let venuesMarker of venue.geojson.features){
127     const koor = venuesMarker.geometry.coordinates;
128     this.userDistanceTo[counter] = this.computeDistance(this.userCoordinatesLat,
129     koor[1],this.userCoordinatesLng,koor[0]);
130     counter+=1;
131   }
132 }
133
134 (function () {
135   initializeMap();
136 })();
137 }
138
139 backToVenue() {
140   this.router.navigate(['venues']);
141 }
142
143 featTapped(venuesCoordinates){
144   const coordinates = venuesCoordinates;
```

```

145     CapacitorGoogleMaps.moveCamera({
146       mapId:this.mapid,
147       cameraPosition:{
148         target:{
149           latitude: coordinates[1],
150           longitude: coordinates[0],
151         },
152         zoom:15
153       },
154       duration:800
155     });
156   }
157
158   computeDistance(lat1, lat2, lon1, lon2){
159
160     const R = 6371e3; // metres
161     const phi1 = lat1 * Math.PI/180; // phi, lambda in radians
162     const phi2 = lat2 * Math.PI/180;
163     const deltaPhi = (lat2-lat1) * Math.PI/180;
164     const deltaLambda = (lon2-lon1) * Math.PI/180;
165
166     const a = Math.sin(deltaPhi/2) * Math.sin(deltaPhi/2) +
167       Math.cos(phi1) * Math.cos(phi2) *
168       Math.sin(deltaLambda/2) * Math.sin(deltaLambda/2);
169     const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
170
171     const d = R * c; // in metres
172
173
174     if(d < 1000){
175       return Math.floor(d) + " m";
176     }else if (d < 100000){
177       return Math.floor(d/1000) + " km";
178     }else{
179       return ">99 km"
180     }
181   }
182 }

```

Kode A.15: venues-map.page.ts

```

1 <ion-header>
2   <ion-toolbar>
3     <ion-buttons slot="start" (click)="backToVenue()">
4       <ion-icon name="arrow-back-outline"></ion-icon>
5     </ion-buttons>
6     <ion-title>Venues</ion-title>
7   </ion-toolbar>
8 </ion-header>
9
10 <div id="container">
11   <div id="mapContainer"></div>
12 </div>
13
14 <!-- //deprecated scroll in ionic 5 -->
15 <ion-content scrollX="true">
16   <ion-label>
17     <h3 #pagetitle id="pagetitle">{{venuesPage.name}}</h3>
18   </ion-label>
19
20   <ion-list>
21     <ion-item *ngFor="let venue of venuesMapsDetail; let i=index">
22       <div>

```

```
23      <div class="venuesDescContainer" *ngFor="let properties of venue.geojson.  
24        features; let j = index">  
25          <div class="venueDesc" (click)="featTapped(properties.geometry.coordinates)">  
26            <h2>{{properties.properties.Name}}</h2>  
27            <p>{{properties.properties.Description}}</p>  
28          </div>  
29          <div class="venuesDist" #distance>  
30            <p>{{userDistanceTo[j]}}</p>  
31          </div>  
32        </div>  
33      </ion-item>  
34    </ion-list>  
35 </ion-content>
```

Kode A.16: venues-map.page.html