BAB II
KAJIAN LITERATUR

2.1 Acara

Acara (*event*) merupakan kegiatan yang dilaksanakan pada waktu tertentu dengan tujuan tertentu dan sifatnya berbeda dengan kegiatan manusia sehari-hari. Pada umumnya acara memiliki hubungan emosional dengan orang yang menghadiri atau mengikutinya [4]. Sebuah acara yang menarik harus memiliki karakteristik dalam penyelenggaraannya, yaitu mempunyai ciri tersendiri dan cenderung memiliki perbedaan antara satu dengan lainnya

Adapun karakteristik acara yang bagus adalah sebagai berikut [5]:

1. *Uniquenesses*

   Kunci utama suksesnya sebuah acara adalah pengembangan ide sehingga acara memiliki keunikan tersendiri. Acara dengan tema yang berbeda, tidak akan mudah untuk dilupakan oleh target *audience*. Keunikan dapat berasal dari peserta yang ikut serta, lingkungan sekitar, pengunjung pada acara tersebut serta beberapa hal lainnya sehingga membuat acara menjadi unik dan berbeda dari yang lainnya.

2. *Perishability*

   Yang dimaksud dengan *perishability* adalah kemungkinan terjadinya acara yang tidak sesuai dengan rencana atau acara tidak hidup sehingga kurang memuaskan. Apabila acara tidak dikemas dengan baik maka target-target yang ingin dicapai di acara tersebut tidak akan tercapai.

3. *Intangibility*

   Setelah menghadiri acara, yang tertinggal di benak pengunjung adalah pengalaman yang mereka dapatkan dari penyelenggaraan acara. Bagi penyelenggara hal ini merupakan tantangan untuk mengubah bentuk pelayanan *intangible* menjadi sesuatu yang berwujud sehingga sekecil apapun wujud yang digunakan dalam acara mampu mengubah persepsi pengunjung. Seperti penggunaan audio visual yang berkualitas yang akan selalu diingat oleh pengunjung acara.

2. *Personal Interaction*

   Personal interaction merupakan salah satu karakteristik yang penting pada saat acara berlangsung. Pengunjung yang datang pada suatu acara juga memiliki peran yang besar terhadap suksesnya acara. Sebagai contoh, keterlibatan aktif penonton pada acara konser

musik dimana penonton dilibatkan untuk bernyanyi sehingga mereka berkontribusi pada terselenggaranya acara tersebut.

2.2 QR Code

Quick Response Code atau yang biasa disebut dengan *QR Code* merupakan sebuah *barcode* dua dimensi yang diperkenalkan oleh Perusahaan Jepang Denso Wave pada tahun 1994. Tipe *barcode* ini awalnya digunakan untuk pendataan inventaris produksi suku cadang kendaraan dan sekarang sudah digunakan dalam berbagai bidang layanan bisnis dan jasa untuk aktivitas marketing dan promosi [6].

Alat pembaca *QR Code* bisa dipasang di *smartphone*, memungkinkan *user* memindai *QR Code* dan melihat data yang disimpan. Untuk menyampaikan informasi dengan cepat dan mendapatkan respons yang cepat, *QR Code* mampu menyimpan informasi secara horizontal dan vertikal, dengan begitu *QR Code* dapat menampung informasi yang lebih banyak daripada *barcode* [7].

2.3 Rapid Application Development

Rapid Application Development (RAD) adalah metodologi untuk mempercepat pengembangan sistem informasi dan menghasilkan sistem informasi yang fungsional. Tujuan utama dari semua pendekatan RAD adalah untuk memangkas waktu pengembangan. Karena tiap tahapan merupakan proses yang berkelanjutan, RAD memungkinkan tim pengembang untuk melakukan perubahan yang diperlukan secara cepat saat terjadi perubahan yang tidak diperkirakan sebelumnya. RAD terdiri dari empat tahapan, yaitu:

1. Perencanaan persyaratan

   Perencanaan persyaratan menggabungkan elemen-elemen dari tahap perencanaan sistem dan analisis sistem pada metodologi *System Development Life Cycle* (SDLC). Pengguna, manajer dan staf TI berdiskusi dan menyepakati ruang lingkup, batasan, dan kebutuhan sistem. Tahapan ini selesai apabila kesepakatan telah tercapai.

2. Desain pengguna

   Di dalam tahapan desain, pengguna berinteraksi dengan analis sistem dan membangun model dan *prototype* yang mewakili semua masukan, proses, dan keluaran sistem. Desain pengguna merupakan proses berkelanjutan dan interaktif yang memberikan pengguna kesempatan untuk memahami, memodifikasi, dan menyetujui model sistem yang memenuhi kebutuhan mereka.

3. Konstruksi sistem

Tahapan konstruksi berfokus pada pengembangan program dan aplikasi. Akan tetapi pada tahapan ini, perubahan atau perkembangan masih bisa diusulkan dan diterapkan.

4. *Cutover*

Fase *cutover* menyerupai tahapan akhir pada fase implementasi di dalam SDLC, seperti konversi data, pengujian, dan pelatihan pengguna [8].

2.4 Teknik Pengembangan Sistem

2.4.1 Value Proposition Canvas

Value proposition pertama kali dikemukakan oleh Alexander Osterwalder. *Value proposition* dalam keseluruhan organisasi, entitas pasar, maupun unit bisnis memungkinkan bisnis untuk memberikan nilai kepada konsumen. *Value proposition* membantu kita untuk menjadi *customer-centric*, yaitu lebih terfokus pada konsumennya. Penerapan *value proposition* dapat menciptakan proposisi penjualan yang menguntungkan. *Value proposition* dilakukan dengan membantu konsumen dalam menyelesaikan permasalahan yang dihadapinya serta membantu konsumen dalam memenuhi kebutuhannya [9].

Value proposition canvas memiliki 2 sisi, yaitu sisi pertama adalah profil pelanggan (*customer profile*), sisi ini digunakan untuk mengklarifikasi pemahaman konsumen. Sedangkan sisi lainnya adalah peta nilai (*value map*), pada sisi ini dijelaskan mengenai bagaimana cara kita untuk menciptakan nilai bagi konsumen. Kemudian setelah mengetahui aspek *customer profile* dan *value map*, maka dapat timbul kecocokan (*fit*) yang dapat muncul ketika kedua aspek tersebut saling bertemu.

1. Profil pelanggan

   Profile Pelanggan membagikan konsumen ke dalam 3 bagian, yaitu tugas konsumen (*customer jobs*), rasa sakit konsumen (*customer pains*), dan sesuatu yang didapatkan konsumen (*customer gains*).

   a. *Customer Jobs*

      Customer Jobs merupakan sesuatu yang ingin diselesaikan oleh konsumen di dalam hidupnya. Setiap konsumen memiliki berbagai tugas yang harus dipenuhi, masalah yang harus diselesaikan, ataupun memenuhi kebutuhan yang harus dipuaskan untuk memastikan kehidupannya dapat berjalan dengan lancar.

   b. *Customer Pains*

      Customer pains menggambarkan situasi yang tidak diinginkan, hambatan-hambatan maupun resiko-resiko yang dihadapi, serta hal-hal negatif lainnya yang

mengganggu konsumen ketika sebelum, selama, maupun setelah konsumen menggunakan produk/jasa yang ditawarkan. Untuk mempermudah kita dalam menganalisis *customer pains*, maka kita dapat melakukan pengukuran tingkat kekecewaan konsumen dengan melihat permasalahan yang dihadapi konsumen ketika menggunakan produk/jasa.

c. *Customer Gains*

Customer gains menggambarkan manfaat/keuntungan yang diharapkan maupun hasil yang ingin dicapai oleh konsumen.

2. *Value Proposition Map*

Setelah mengidentifikasi *customer profile*, maka kita dapat menentukan *value proposition map. Value proposition map* digunakan untuk mengatasi permasalahan yang dihadapi oleh konsumen. *Value Proposition Map* dapat dibagi menjadi 3 aspek, yaitu:

a. *Product & Services*

Menjelaskan bahwa aspek ini merupakan keseluruhan produk dan jasa yang ditawarkan. Pada aspek ini, kita memberikan daftar semua produk dan jasa yang ingin dibangun dari *value proposition* yang ditawarkan dengan kebutuhan konsumen. Produk dan jasa yang ditawarkan harus dapat membantu untuk memenuhi kebutuhan fungsional, sosial, emosional dan memuaskan konsumen dalam memenuhi kebutuhan dasarnya.

b. *Pain Relievers*

Setelah menetapkan *customer pains*, kita perlu untuk menentukan *pain relievers* sebagai tindakan solusi untuk mengatasi permasalahan-permasalahan yang akan dihadapi oleh calon konsumen. *Pain relievers* menggambarkan bagaimana produk dan jasa dapat meringankan/mengatasi *customer pains* secara spesifik.

c. *Gain Creators*

Gain creators menggambarkan bagaimana produk dan jasa dapat menciptakan *customer gains*. Dalam menetapkan *gain creators*, kita harus mampu menciptakan manfaat/keuntungan kepada konsumen sesuai dengan harapan dan keinginan dari konsumen, seperti memberikan manfaat fungsional, keuntungan sosial, emosi yang positif, serta penghematan biaya. *Gain creators* membantu konsumen dalam memenuhi kebutuhannya, dan menjawab permasalahan yang dihadapi oleh konsumen [10].

2.4.2 Use Case Diagram

Use Case Diagram merupakan diagram yang menggambarkan hubungan interaksi antara sistem dengan sistem eksternal dan juga pengguna, dengan kata lain, *use case* ini menggambarkan siapa yang akan menggunakan sistem dan dengan cara bagaimana pengguna dapat saling berinteraksi dengan sistem.

Berikut ini adalah elemen-elemen yang digunakan dalam diagram *use-case* [11]:

1. *Actor*

   Mempresentasikan seseorang atau sesuatu (seperti perangkat, sistem lain) yang berinteraksi dengan sistem. *Actor* hanya berinteraksi dengan *use case* tetapi tidak memiliki kontrol atas *use case*.

2. *Use Case*

   Gambaran fungsionalitas dari suatu sistem, sehingga pengguna sistem paham dan mengerti mengenai kegunaan sistem yang akan dibangun.

3. Relasi

   Merupakan hubungan yang terjadi pada sistem baik antar aktor maupun antar *use case* maupun antara *use case* dan aktor. Relasi yang digunakan dalam diagram *use case* antara lain:

   a. *Assosiation*, merupakan relasi yang digunakan untuk menggambarkan interaksi antara *use case* dan aktor.

   b. *Generalization*, merupakan relasi yang menggambarkan *inheritance* baik aktor maupun *use case*.

   c. *Dependency*, merupakan relasi yang menggambarkan ketergantungan antara *use case* yang satu dengan *use case* yang lain. Ada dua macam *dependency* yaitu *include* dan *extends*. *Include* menggambarkan bahwa jalannya suatu *usecase* memicu jalannya *usecase* lainnya. Sedangkan *extends* menggambarkan bahwa suatu *usecase* dijalankan karena ada persyaratan tertentu dari *usecase* lainnya.

2.4.3 PIECES

Untuk menganalisis persyaratan non fungsional apa saja yang ada didalam sistem, maka dilakukan analisis terhadap kinerja, informasi, ekonomi, pengendalian, efisiensi, dan pelayanan. Metode ini dikenal dengan analisis PIECES (*performance*, *information*, *economic*, *control*, *efficiency*, *service*).

Berikut penjelasan mengenai masing-masing komponen dari PIECES [12]:

1. Analisis kinerja sistem (*performance*)

   Kinerja adalah suatu kemampuan sistem dalam menyelesaikan tugas dengan cepat sehingga sasaran dapat segera tercapai. Kinerja diukur dengan jumlah produksi (*throughput*) dan waktu yang digunakan untuk menyesuaikan perpindahan pekerjaan (*response time*).

2. Analisis informasi (*information*)

   Informasi merupakan hal penting karena dengan informasi tersebut pihak manajemen (*marketing*) dan *user* dapat melakukan langkah selanjutnya. Apabila kemampuan sistem informasi baik, maka *user* akan mendapatkan informasi yang akurat, tepat waktu dan relevan sesuai dengan yang diharapkan.

3. Analisis ekonomi (*economy*)

   Pemanfaatan biaya yang digunakan dari pemanfaatan informasi. peningkatan terhadap kebutuhan ekonomis mempengaruhi pengendalian biaya dan peningkatan manfaat.

4. Analisis pengendalian (*control*)

   Analisis ini digunakan untuk membandingkan sistem yang dianalisa berdasarkan pada segi ketepatan waktu, kemudahan akses, dan ketelitian data yang diproses.

5. Analisis efisiensi (*efficiency*)

   Efisiensi berhubungan dengan bagaimana sumber tersebut dapat digunakan secara optimal. Operasi pada suatu perusahaan dikatakan efisien atau tidak biasanya didasarkan pada tugas dan tanggung jawab dalam melaksanakan kegiatan.

6. Analisis pelayanan (*service*)

   Peningkatan pelayanan memperlihatkan kategori yang beragam. Proyek yang dipilih merupakan peningkatan pelayanan yang lebih baik bagi manajemen (*marketing*), *user* dan bagian lain yang merupakan simbol kualitas dari suatu sistem informasi.

2.5 Aplikasi Seluler dan Aplikasi Berbasis Web

Aplikasi seluler (*Mobile Apps*) yaitu aplikasi yang dibuat untuk perangkat-perangkat bergerak (*Mobile*) seperti: *Smartphone*, *SmartWatch*, *Tablet*, dan lainnya. Perangkat lunak atau disebut juga *software* aplikasi merupakan hasil dari pemrograman yang dirancang menggunakan bahasa pemrograman tertentu [13]. Selain itu, penggunanya harus melakukan pengunduhan atau download serta mengunduh dari toko aplikasi seperti *Google Play Store*, *Apple App Store* dan masih banyak lagi toko aplikasi yang sesuai dengan *platform* aplikasi seluler [14].

2.6 Pengembangan Front-End dan Back-End Aplikasi

Front-end merupakan istilah yang digunakan untuk pengembang yang bertugas untuk membuat tampilan dalam bentuk situs web ataupun aplikasi seluler menggunakan bahasa pemrograman. Tampilan ini merupakan tampilan yang akan dilihat dan dioperasikan secara langsung oleh pengguna.

Pada umumnya pengerjaan *front-end* dibagi menjadi 2 bagian. Yang pertama adalah UI / UX Designer. Pada tahap tersebut, UI/UX Designer bertugas untuk membuat sebuah desain atau rancangan awal pembuatan *website*. Kemudian setelah membuat desain awal / *prototype*, seorang *front-end developer* bertugas untuk menerapkan dan menerjemahkan desain tersebut ke dalam bentuk bahasa pemrograman [15].

Back-end atau sering disebut *server side* adalah tempat dimana proses sebuah aplikasi atau sistem berjalan, proses di *back-end* biasanya digunakan untuk menambahkan, mengubah atau menghapus data. *Back-end* biasanya tidak langsung berinteraksi kepada *user*, yaitu seperti *database* dan *server*. Biasanya orang yang bekerja sebagai *back-end developer* adalah *programmer* atau *developer* yang fokus pekerjaannya pada keamanan, desain sistem, dan manajemen data pada sistem. *Back-end developer* dibutuhkan dalam pengembangan sistem atau aplikasi dinamis yang memiliki data yang selalu berubah ubah, contoh *website* dinamis antara lain *facebook* dan *google*. Seorang *back-end developer* biasanya harus menguasai bahasa pemrograman yang dapat digunakan untuk mengelola *database*, mengolah file dan I/O seperti PHP, ASP, dan *NodeJs* [16].

2.6.1 React-Native

React-Native merupakan kerangka kerja *JavaScript* yang digunakan untuk membangun aplikasi seluler *Android* maupun *iOS*. *React-Native* ini memiliki dasar dari *React* dan *library JavaScript* dalam membangun antarmuka. *React-Native* ini ditulis dengan campuran *JavaScript* dan *JSX*, lalu *React-Native* ini juga memaparkan antarmuka *JavaScript* untuk *platform API* dimana pengembang dalam membangun aplikasi ini dapat mengakses fitur-fitur seperti kamera, lokasi, dan lainnya yang ada pada *smartphone* [17].

2.6.2 ReactJS

ReactJS atau React merupakan *open-source Javascript library* untuk mengembangkan antarmuka pengguna yang lebih interaktif dan mempermudah *developer* dalam perancangan aplikasi. ReactJS berusaha untuk memberikan kecepatan, kesederhanaan, dan skalabilitas. Beberapa fitur ReactJS yang biasa dikenal adalah *JSX* atau *Javascript XML*. *JSX* adalah

extension untuk sintaksis *ECMAScript*. *JSX* membantu *developer* pada saat mengembangkan UI di dalam *Javascript*, dan juga dapat membantu *developer* ketika sedang melakukan *error debugging* [18].

2.6.3 API

API adalah singkatan dari *Application Programming Interface* yang merupakan perantara antar perangkat lunak. *API* berisi sekumpulan definisi dan protokol untuk membuat dan mengintegrasikan perangkat lunak. Dalam perangkat lunak dengan arsitektur *client-server*, *API* dapat kita analogikan sebagai pelayan, dengan *client* sebagai pengunjung dan *server* sebagai dapur. Tugas dari *API* adalah menjembatani antara pengunjung dan dapur, dengan menerima pesanan dari pengunjung lalu menyampaikannya ke dapur (*request*). Setelah pesanan berhasil dibuat, pelayan kemudian mengantarkan makanan dari dapur kepada pengunjung yang memesan (*response*). *API* juga seringkali disebut sebagai sebuah kontrak, yang berisi dokumentasi yang berisi kesepakatan antara pihak-pihak yang terlibat. Pihak 1 harus mengirim *request* terstruktur dengan cara yang telah ditentukan, dan Pihak 2 akan mengirim *response* dengan cara yang telah disepakati [19].

2.6.4 Node.js

Node.js adalah adalah *runtime JavaScript* yang dibangun di mesin *JavaScript V8 Chrome*. Node.js menggunakan model I / O event-*driven*, yang membuatnya ringan dan efisien. Ekosistem paket Node.js, npm merupakan ekosistem perpustakaan *open source* terbesar di dunia. Node.js adalah termasuk bahasa *javascript* dimana bahasa pemrograman ini sering digunakan dalam development [20].

2.7 Basis Data

Perkembangan teknologi informasi menuju ke arah digitalisasi data dengan menggunakan sistem basis data komputer dikarenakan kebutuhan pengolahan data yang cepat dan efisien. Basis data dimaksudkan untuk mengatasi *problem* pada sistem yang memakai pendekatan barbasis berkas. *Database* diimplementasikan dalam sebuah perangkat lunak untuk memanajemen *database* tersebut. Perangkat lunak yang digunakan untuk manajemen *database* adalah DBMS (*Database Management System*). *Database Management System* (DBMS) adalah kumpulan program yang digunakan untuk mendefinisikan, mengatur, dan memproses *database*. Sedangkan *database* esensinya adalah

sebuah struktur yang dibangun untuk keperluan penyimpanan data. Tujuan utama DBMS adalah untuk menyediakan tinjauan abstrak dari data bagi *user* [21]. DBMS meliputi

1. Sebuah *modeling language* untuk mendefinisikan skema (*relational model*) dari setiap *database* yang berada di DBMS sesuai dengan data modelnya. Pemilihan struktur yang paling cocok tergantung aplikasi, kecepatan transaksi dan banyak model.
2. Struktur data (*field*, *record* dan *file*) dioptimalkan dan disesuaikan dengan kebutuhan penyimpanan data di sebuah media penyimpanan yang permanen.
3. Mekanisme transaksi yang idealnya tetap menjaga integritas data walaupun akses dilakukan oleh banyak pemakai secara bersamaan [22].

MySQL adalah sistem manajemen basis data relasi yang bersifat terbuka atau *open source*. Tujuan awal ditulisnya program MySQL adalah untuk mengembangkan aplikasi Web. MySQL menggunakan bahasa standar SQL (*Structure Query Language* sebagai bahasa interaktif dalam mengelola data. Perintah SQL sering juga disebut *Query* [23].