BAB 2
TINJAUAN PUSTAKA

2.1 Ruang Lingkup Objek Penelitian

Objek penelitian yang sedang diteliti adalah Pesona Nirwana waterpark, yang bertempat di Kp. Legok Jeungjing, Desa Panyirapan Kec. Soreang – Kabupaten Bandung.

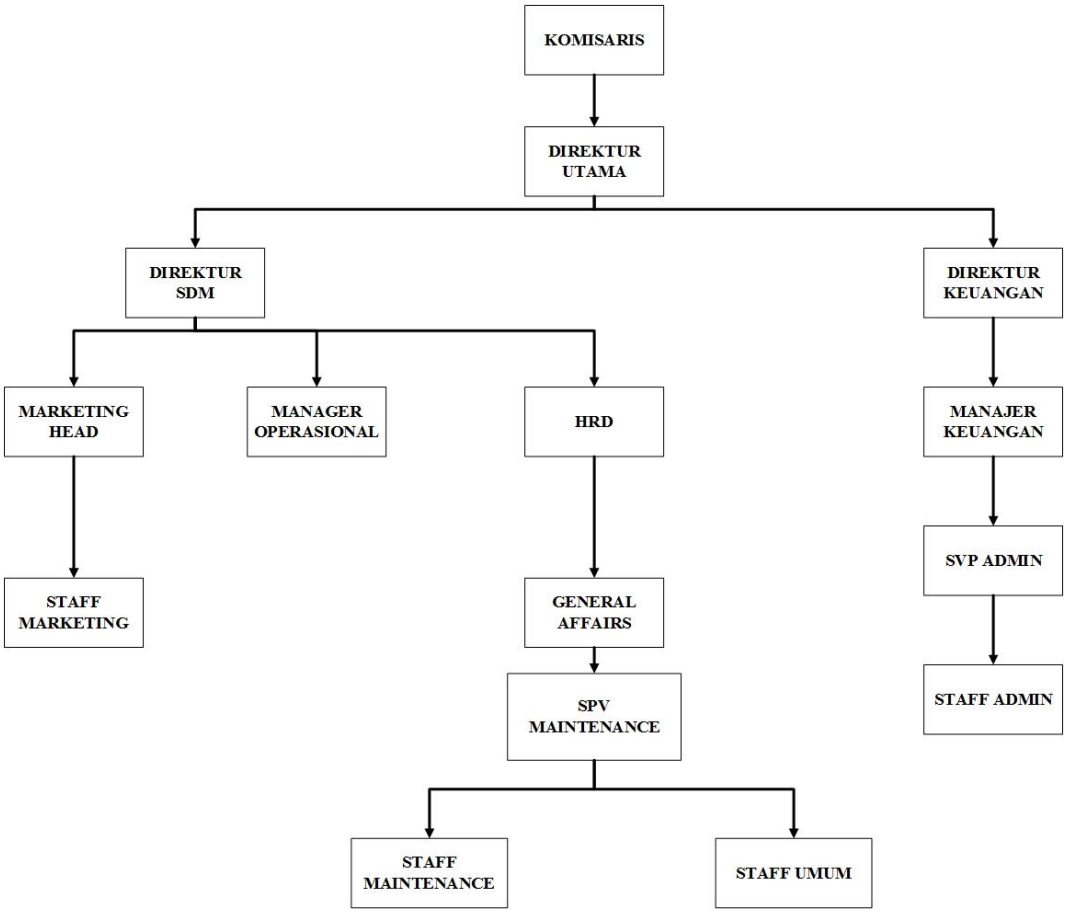
2.1.1 Sejarah Pesona Nirwana *Waterpark*

Pesona Nirwana Waterpark didirikan pada tahun 2013 oleh PT. Wiyasa, sebuah konsultan perdagangan umum. Satu-satunya wahana permainan air bersama keluarga yang terdapat di Kp. Legok Jeungjing Desa Panyirapan Kecamatan Soreang – Kabupaten Bandung yang memiliki luas 4000 meter persegi. Keunikan tempat ini adalah berupa waterpark yang dikelilingi oleh dinding bebatuan keras setinggi 20 meter menambah keindahan panorama alam.

Wahana yang ditawarkan di Pesona Nirwana *Waterpark* sangat beragam sekali mulai dari wahana untuk remaja/dewasa hingga untuk anak-anak, diantaranya ada wahana *Lazy River, Kiddy Pool, Water Slider, Jacuzzi*. Selain wahana bermain Pesona Nirwana Waterpark pun memiliki foodcourt yang menyediakan banyak makanan dan minuman. Berbagai makanan western pun dihidangkan di foodcourt Pesona Nirwana seperti pizza, burger, *spaghetti* yang bisa disantap oleh pengunjung setelah lelah bermain air.

2.1.2 Struktur Organisasi

Struktur organisasi Pesona Nirwana Waterpark dapat dilihat pada gambar 2.1



Gambar 2. 1 Struktur Organisasi

2.2 Landasan Teori

Landasan teori merupakan kumpulan dari beberapa definisi dan konsep yang disusun secara sistematis mengenai variabel-variabel dalam penelitian. Landasan teori ini akan menjadi acuan dasar penulis dalam melakukan penelitian melalui uraian-uraian secara teoritis.

2.2.1 Sistem Pembayaran

Sistem pembayaran adalah sistem yang mencakup seperangkat aturan, lembaga, dan mekanisme yang dipakai untuk melaksanakan pemindahan dana guna memenuhi suatu kewajiban yang timbul dari suatu kegiatan ekonomi. Lantas, apa saja komponen dari Sistem pembayaran ? Sudah barang tentu harus ada alat pembayaran, ada mekanisme kliring hingga penyelesaian akhir (*settlement*). Selain itu juga ada komponen lain seperti lembaga yang terlibat dalam menyelenggarakan sistem pembayaran. Termasuk dalam hal ini adalah bank, lembaga keuangan selain bank, lembaga bukan bank penyelenggara transfer dana, perusahaan switching bahkan hingga bank sentral [3].

2.2.1.1 Evolusi Alat Pembayaran

Alat pembayaran boleh dibilang berkembang sangat pesat dan maju. Kalau kita menengok kebelakang yakni awal mula alat pembayaran itu dikenal, sistem barter antarbarang yang diperjualbelikan adalah kelaziman di era pra moderen. Dalam perkembangannya, mulai dikenal satuan tertentu yang memiliki nilai pembayaran yang lebih dikenal dengan uang. Hingga saat ini uang masih menjadi salah satu alat pembayaran utama yang berlaku di masyarakat. Selanjutnya alat pembayaran terus berkembang dari alat pembayaran tunai (*cash based*) ke alat pembayaran nontunai (*non cash*) seperti alat pembayaran berbasis kertas (*paper based*), misalnya, cek dan bilyet giro. Selain itu dikenal juga alat pembayaran paperless seperti transfer dana elektronik dan alat pembayaran memakai kartu (card-based) (ATM, Kartu Kredit, Kartu Debit dan Kartu Prabayar) [3].

2.2.1.2 Alat Pembayaran Tunai

Alat pembayaran tunai lebih banyak memakai uang kartal (uang kertas dan logam). Uang kartal masih memainkan peran penting khususnya untuk transaksi bernilai kecil. Dalam masyarakat moderen seperti sekarang ini, pemakaian alat pembayaran tunai seperti uang kartal memang cenderung lebih kecil dibanding uang giral. Pada tahun 2005, perbandingan uang kartal terhadap jumlah uang beredar sebesar 43,3 persen.

Namun patut diketahui bahwa pemakaian uang kartal memiliki kendala dalam hal efisiensi. Hal itu bisa terjadi karena biaya pengadaan dan pengelolaan (*cash handling*) terbilang mahal. Hal itu belum lagi memperhitungkan inefisiensi dalam waktu pembayaran. Misalnya, ketika Anda menunggu melakukan pembayaran di loket pembayaran yang relatif memakan waktu cukup lama karena antrian yang panjang. Sementara itu, bila melakukan transaksi dalam jumlah besar juga mengundang risiko seperti pencurian, perampokan dan pemalsuan uang. Menyadari ketidak-nyamanan dan inefisien memakai uang kartal, BI berinisiatif dan akan terus mendorong untuk membangun masyarakat yang terbiasa memakai alat pembayaran nontunai atau *Less Cash Society* (LCS) [3].

2.2.1.3 Alat Pembayaran Nontunai

Alat pembayaran nontunai sudah berkembang dan semakin lazim dipakai masyarakat. Kenyataan ini memperlihatkan kepada kita bahwa jasa pembayaran nontunai yang dilakukan bank maupun lembaga selain bank (LSB), baik dalam proses pengiriman dana, penyelenggara kliring maupun sistem penyelesaian akhir (*settlement*) sudah tersedia dan dapat berlangsung di Indonesia. Transaksi pembayaran nontunai dengan nilai besar diselenggarakan Bank Indonesia melalui sistem BI-RTGS (*Real Time Gross Settlement*) dan Sistem Kliring. Sebagai informasi, sistem BI-RTGS adalah muara seluruh penyelesaian transaksi keuangan di Indonesia.

Bisa dibayangkan, hampir 95 persen transaksi keuangan nasional bernilai besar dan bersifat mendesak (*urgent*) seperti transaksi di Pasar Uang Antar Bank (PUAB), transaksi di bursa saham, transaksi pemerintah, transaksi valuta asing (valas) serta settlement hasil kliring dilakukan melalui sistem BI-RTGS. Pada tahun 2010, BI-RTGS melakukan transaksi sedikitnya Rp174,3 triliun per hari. Sedangkan transaksi nontunai dengan alat pembayaran menggunakan kartu (APMK) dan uang elektronik masing-masing nilai transaksinya hanya Rp8,8 triliun per hari yang dilakukan bank atau LSB.

Melihat pentingnya peran BI-RTGS dalam sistem pembayaran nasional, sudah barang tentu harus dijaga kontinuitas dan stabilitasnya. Bila sesaat saja

sistem BI-RTGS ini ngadat atau mengalami gangguan jelas akan sangat menganggu kelancaran dan stabilitas sistem keuangan di dalam negeri. Hal itu belum memperhitungkan dampak material dan nonmaterial dari macetnya sistem BI-RTGS tadi. Untuk itulah BI sangat peduli menjaga stabilitas BI-RTGS yang dikategorikan sebagai Systemically Important Payment System (SIPS). SIPS adalah sistem yang memproses transaksi pembayaran bernilai besar dan bersifat mendesak (urgent).Adalah wajar saja apabila Bank Indonesia sangat peduli menjaga kestabilan SIPS dengan mengelola risiko, desain, kehandalan teknologi, jaringan pendukung dan aturan main dalam SIPS. Selain SIPS dikenal pula System Wide Important Payment System (SWIPS), yaitu sistem yang digunakan oleh masyarakat luas. Sistem Kliring dan APMK termasuk dalam kategori SWIPS ini. BI juga peduli dengan SWIPS karena sifat sistem yang digunakan secara luas oleh masyarakat. Apabila terjadi gangguan maka kepentingan masyarakat untuk melakukan pembayaran akan terganggu pula, termasuk kepercayaan terhadap sistem dan alat-alat pembayaran yang diproses dalam sistem.

Perlu diketahui bahwa BI bukan semata peduli akan terciptanya efisiensi dalam sistem pembayaran, tapi juga kesetaraan akses hingga ke urusan perlindungan konsumen. Yang dimaksud terciptanya sistem pembayaran, itu artinya memberi kemudahan bagi pengguna untuk memilih metode pembayaran yang dapat diakses ke seluruh wilayah dengan biaya serendah mungkin. Sementara yang dimaksud dengan kesetaraan akses, BI akan memperhatikan penerapan asas kesetaraan dalam penyelenggaraan sistem pembayaran. Sedangkan aspek perlindungan konsumen dimaksudkan penyelenggara wajib mengadopsi asas-asas perlindungan konsumen secara wajar dalam penyelenggaraan sistemnya [3].

2.2.2 Jaringan Komputer

Jaringan komputer adalah sekumpulan komputer yang saling terhubung. Tujuan utama dari jaringan komputer ini yaitu untuk pertukaran data dan informasi antar komputer dalam satu jaringan.

Keuntungan yang didapat dengan menggunakan jaringan komputer diantaranya :

a. Dapat mengakses data di komputer lain dari komputer yang anda gunakan.
b. Data yang digunakan dapat disimpan atau disalin ke beberapa komputer, sehingga bila salah satu komputer rusak maka salinan di komputer lain dapat digunakan.
c. Penggunaan printer, *scanner*, CD/DVD ROM dan perangkat lainnya dapat digunakan bersama-sama dengan pengguna lain.
d. Administrator jaringan dapat mengontrol data-data penting agar dapat diakses oleh pengguna yang berhak saja.
e. Penghematan biaya dapat dilakukan, karena sebuah perangkat dapat digunakan bersama-sama.

Berdasarkan jangkauan area atau lokasi, jaringan dibedakan menjadi 3 jenis yaitu :

a. *Local Area Network* (LAN)

   Local Area Network (LAN) merupakan jaringan yang menghubungkan sejumlah komputer yang ada dalam suatu lokasi dengan area yang terbatas seperti ruang atau gedung. LAN dapat menggunakan media komunikasi seperti kabel dan *wireless*.

b. *Wide Area Network* (WAN)

   Wide Area Network (WAN) merupakan jaringan antara LAN satu dengan lain yang dipisahkan oleh lokasi yang cukup jauh. Contoh penggunaan WAN adalah hubungan antara kantor pusat dengan kantor cabang yang ada di daerah-daerah.

c. *Metropolitan Area Network* (MAN)

Metropolitan Area Network (MAN) merupakan jaringan yang lebih besar dari jaringan LAN tetapi lebih kecil dari jaringan WAN. Jaringan MAN dan WAN sama-sama menghubungkan beberapa LAN yang membedakan hanya lingkup areanya yang berbeda.

Berdasarkan fungsinya jaringan komputer terbagi menjadi dua jenis yaitu:

a. Jaringan *Client Server*

Jaringan *client server* menghubungkan komputer server dengan komputer klien. Komputer server adalah komputer yang menyediakan fasilitas bagi komputer-komputer klien yang terhubung dalam jaringan. Sedangkan komputer klien adalah komputer yang menggunakan fasilitas yang disediakan komputer server. Komputer server pada sebuah jaringan client server disebut sebagai *Dedicated Server*, karena komputer yang digunakan hanya sebagai penyedia fasilitas komputer klien. Komputer server tidak dapat berperan sebagai komputer klien.

Keunggulan menggunakan jaringan *client server* yaitu :

1. Terdapat administrator jaringan yang mengelola sistem keamanan dan administrasi jaringan.
2. Komputer server difungsikan sebagai pusat data.
3. Pengaksesan data lebih tinggi karena penyediaan dan pengelolaan fasilitas jaringan dilakukan oleh komputer server.
4. Sistem *backup* data lebih baik, karena *backup* data dapat dilakukan terpusat di komputer server.

Kelemahan menggunakan jaringan *client server* yaitu :

1. Biaya mahal, karena membutuhkan komputer yang memiliki kemampuan tinggi yang difungsikan sebagai komputer server.
2. Kelancaran jaringan bergantung pada komputer server, bila komputer server mengalami gangguan maka jaringan akan terganggu.

b. Jaringan *Peer To Peer*

Jaringan *Peer To Peer* menghubungkan beberapa komputer dalam sebuah jaringan. Pertukaran data dapat dilakukan antar komputer yang

terhubung tanpa perantara komputer server. Masing-masing komputer dapat berperan sebagai komputer server sekaligus sebagai komputer klien.

Keunggulan menggunakan jaringan *peer to peer* yaitu :

1. Semua komputer yang terhubung memiliki hak yang sama.
2. Biaya lebih murah karena tidak memerlukan adanya sebuah komputer server.
3. Kelancaran jaringan tidak bergantung pada komputer server.

Kelemahan menggunakan jaringan *peer to peer* yaitu :

1. Troubleshooting jaringan lebih rumit
2. Sistem keamanan jaringan ditentukan oleh masing-masing pengguna dengan mengatur keamanan pada fasilitas yang dimiliki.
3. *Backup* data dilakukan pada masing-masing komputer [4].

2.2.3 Intranet

Intranet adalah konsep LAN yang mengadopsi teknologi Internet dan mulai diperkenalkan pada akhir tahun 1995. Dengan kata lain intranet adalah LAN yang menggunakan standar komunikasi dan segala fasilitas internet. Intranet dapat diibaratkan berinternet dalam lingkungan local dan pada umumnya terkoneksi ke internet sehingga memungkinkan pertukaran informasi dan data dengan jaringan intranet lainnya melalui *backbone* Internet.

Intranet adalah sebuah jaringan privat (*private network*) yang menggunakan protokol-protokol Internet (TCP/IP) untuk membagi informasi rahasia perusahaanatau operasi dalam perusahaan tersebut kepada karyawannya. Kadang-kadang, istilah intranet hanya merujuk kepada layanan yang terlihat, yakni situs web internal perusahaan. Untuk membangun sebuah intranet maka sebuah jaringan harus memiliki beberapa komponen yang membangun Internet yaitu protokol Internet (Protokol TCP/IP, alamat IP, dan protokol lainnya), *client* dan juga *server*. Protokol HTTP dan beberapa protokol Internet lainnya (FTP, POP3, atau SMTP) umumnya merupakan komponen protokol yang sering digunakan.

Umumnya, sebuah intranet dapat dipahami sebagai sebuah versi pribadi dari jaringan internet atau sebagai sebuah versi dari Internet yang dimiliki oleh sebuah organisasi. Adapun manfaat dari Intranet adalah :

1. Membuat perusahaan atau institusi menjadi lebih hemat dan efisien, karena sistem informasi berubah dari cara konvensional ke cara yang memanfaatkan web dan *database*.
2. Perolehan informasi yang cepat dan detail serta sistem manajemen yang canggih akan membuat perusahaan atau intsitusi tersebut memimpin dalam bidangnya.
3. Pertukaran data antar lokasi secara *point to point*, seperti *video conference*, transfer data atau file dan telepon intranet [5].

2.2.4 Pemodelan Analisis

Model analisis merupakan representasi teknis yang pertama dari sistem. Pertama analisis terstruktur adalah metode pemodelan klasik, dan analisis berorientasi objek.

Analisis terstruktur merupakan salah satu teknis analisis yang menggunakan pendekatan berorientasi fungsi. Teknik ini mempunyai sekumpulan perangkat yang memungkinkan analisis sistem mendefinisikan spesifikasi fungsional perangkat lunak secara terstruktur, yang dimaksud dengan perangkat analisis terstruktur adalah alat bantu pemodelan yang digunakan untuk menggambarkan hasil pelaksanaan analisis.

Entity-relational diagram (ERD) adalah notasi yang digunakan untuk melakukan aktivitas pemodelan data, sedangkan *data flow diagram* (DFD) memberikan informasi sebagai dasar bagi pemodelan fungsi.

2.2.5 Basis Data

Basis data adalah kumpulan data yang saling berkaitan, berhubungan yang disimpan secara bersama-sama sedemikian rupa tanpa pengulangan yang tidak perlu, untuk memenuhi berbagai kebutuhan. Data ini mengandung semua informasi untuk mendukung semua kebutuhan sistem.

Basis data merupakan salah satu komponen yang penting dalam sebuah sistem, karena merupakan basis dalam menyediakan informasi. Basis data menjadi penting karena munculnya beberapa masalah bila tidak menggunakan data terpusat, seperti adanya duplikasi data, hubungan antar data tidak jelas, organisasi data dan perbaharuan data menjadi rumit. Bahasa yang digunakan dalam basis data yaitu [6].

1. DDL (*Data Definition Language*)

   Merupakan bahasa definisi data yang digunakan untuk membuat dan mengelola objek *database* seperti *database*, tabel, *view.* Adapun perintah yang dimiliki adalah :

   a. CREATE : Perintah untuk pembuatan tabel dan *database.*
   b. DROP : Perintah untuk penghapusan tabel dan *database.*
   c. ALTER : Perintah untuk pengubahan struktur tabel yang telah dibuat, seperti menambahkan atribut, mengganti nama atribut ataupun merubah nama tabel.

2. DML (*Data Manipulation Language*)

   Merupakan bahasa manipulasi data yang dilakukan setelah DDL, digunakan untuk memanipulasi data pada objek *database* seperti tabel yang telah dibuat sebelumnya, adapun perintah yang dimiliki DDL adalah :

   a. INSERT : untuk melakukan penginputan atau pemasukan data pada tabel *database.*
   b. UPDATE : : Untuk melakukan perubahan/peremajaan terhadap data yang ada pada tabel.
   c. DELETE : Untuk melakukan penghapusan data pada tabel. Penghapusan ini dapat dilakukan secara sekaligus (seluruh isi tabel) maupun hanya beberapa *recordset*.

3. DCL (*Data Control Language*)

   Merupakan bahasa yang digunakan untuk mengendalikan pengaksesan data pada *database,* adapun perintah yang dimiliki DCL adalah:

a. GRANT : Untuk memberikan hak akses atau izin oleh *administrator server* kepada pengguna.
b. REVOKE : Untuk menghilangkan atau mencabut hak akses yang telah diberikan kepada *user* oleh *administrator*.

Pengolahan basis data meliputi proses ketiga bahasa yang digunakan diatur dengan menggunakan perangkat sistem manajemen basis data (*Database Management Sistem* / DBMS).

2.2.6 Perangkat Lunak

Perangkat lunak (*software*) adalah program komputer yang terasosiasi dengan dokumentasi perangkat lunak seperti dokumentasi kebutuhan, model desain, dan cara penggunaan (*user manual*). Sebuah program komputer tanpa terasosiasi dengan dokumentasinya maka belum dapat disebut perangkat lunak (*software*). Sebuah perangkat lunak juga sering disebut dengan sistem perangkat lunak. Sistem berarti kumpulan komponen yang saling terkait dan mempunyai satu tujuan yang ingin dicapai.

Sistem perangkat lunak berarti sebuah sistem yang memiliki komponen berupa perangkat lunak yang memiliki hubungan satu sama lain untuk memenuhi kebutuhan pelanggan (*customer*). Pelanggan (*customer*) adalah orang atau organisasi yang memesan atau membeli perangkat lunak (*software*) dari pengembang perangkat lunak atau bisa dianggap bahwa pelanggan (*customer*) adalah orang atau organisasi yang dengan sukarela mengeluarkan untuk memesan atau membeli perangkat lunak. *User* atau pemakai perangkat lunak adalah orang yang memiliki kepentingan untuk memakai atau menggunakan perangkat lunak untuk memudahkan pekerjaannya [8].

Secara umum perangkat lunak ini dapat diklasifikasikan menjadi 2, yaitu:

1. Perangkat Lunak Sistem Operasi

Merupakan perangkat lunak yang berfungsi untuk mengkonfigurasikan komputer agar dapat menerima berbagai perintah dasar yang diberikan sebagai masukan. Perangkat lunak sistem juga merupakan sekumpulan program yang ditulis untuk melayani program-program yang lain. Contoh: *MS-DOS, LINUX, UNIX, FREE*

BSD, OS/2, SUN OS (JAVA), WINDOWS, MACINTOSH, NOVELL, dan lain sebagainya.

2. Perangkat Lunak Aplikasi

Merupakan program siap pakai yang digunakan untuk aplikasi dibidang tertentu. Perangkat lunak aplikasi yang membantu *user* sehingga dapat bekerja lebih efektif dan efisien. Aplikasi ini dibagi atas beberapa bidang, antara lain:

1) *Business & Office Application*

   Aplikasi perkantoran dipergunakan untuk menyelesaikan pekerjaan kantor seperti mengetik (*Word processing, document management*) membuat tabel kerja (*Worksheet*), membuat Database sederhana (*MS. Access*) ataupun mengolah *image*/citra sederhana. Contoh aplikasi perkantoran yang terpaket komplit: *Mocrosoft Office, Koffice, Star Office, Abiword, Open Office, MS_Office for MAC,* dan lain sebagainya. Aplikasi bisnis lainnya juga meliputi aplikasi pengolahan data keuangan, misal (*Personal Finance Application & Business Acounting Application*). Contoh: Aplikasi perhitungan neraca, rugi laba perusahaan, Aplikasi perhitungan pajak personal & perusahaan, Aplikasi manajemen proyek, Aplikasi persediaan stok barang dan penjualan, dan lain sebagainya.

2) *Database Application*

   Aplikasi yang digunakan dalam pengolahan data baik yang berukuran kecil maupun besar bisa digunakan secara *stand-alone* (tunggal) maupun dalam sistem berbasis jaringan lokal *Client server* maupun *webbase* (intranet maupun internet). Contoh: *MS SQL, FoxBase, Oracle, MySQL, PostgresSQL, SQL Server*, dan lain sebagainya.

3) *Graphics Design Application*

   Aplikasi yang digunakan untuk membuat sketsa dua dimensi untuk desain ruangan, perabotan, mesin-mesin model industri, juga model

tubuh manusia, hewan, dan lain sebagainya. Contoh: *AutoCad, Pro Design, CorelDraw, Adobe Ilistrator,* dan lain sebagainya.

Ada juga yang memasukan dalam golongan ini aplikasi *image, audio, video manipulator*, yaitu aplikasi yang digunakan untuk mengolah citra, suara hingga gambar 3D untuk pembuatan animasi, pengolahan *image, photo, audio, video editing*, dan lain sebagainya. Contoh: *Adobe (Photoshop, After Effect, Premire, & Audition), Corel (Draw, RAVE, Photo Paint), Ulead Media studio, GIMP (In Linux), Cool Edit Pro, Sound Force, Macromedia (Flash, Director), 3D Studio Max*, dan lain sebagainya.

4) *Antivirus & Utility Tools*

Software ini digunakan untuk operasi minor (*Background*) yang bekerja di belakang sistem, berguna untuk meningkatkan kinerja sistem operasi atau aplikasi dan juga performa *hardware*. Terdapat 2 jenis utilitas di dalam sistem operasi:

a) *Utilitas Internal*, sudah termasuk dalam paket yang di instalasi dalam sistem operasi seperti *scandisk* di *Windows* atau perintah *fsck* (*file system check*) di OS Linux.
b) *Utilitas eksternal*, yang harus diinstalasi kemudian pada OS, seperti *System Optimizer, PC Tools, Data/File Recovery, PC Maintenance, Internet Utilities, Norton System Work, Registry Clean & Repair, Hardware Repair*.

Sementara untuk proteksi virus dan antivirus, contohnya adalah antivirus produk *Mc Affee, Grisoft, Norton, PandaSoft,* dan lain sebagainya.

5) *Development Tools*

Merupakan sistem bahasa pemrograman komputer yang dilengkapi dengan *compiler* tertentu sebagai sumber media penyusunan program aplikasi. Contoh: *QuikBasic, Clipper, Visual Basic, Turbo Pascal,*

Visual C++, GCC, Turbo Assembler, Phyton, KDEvelopt, Gdekstop, dan lain sebagainya.

Termasuk dalam jenis aplikasi ini adalah *web development tools*, yaitu aplikasi yang digunakan untuk membuat/mendesain tampilan website (*homepage*) di internet, seperti *FrontPage, Macromedi Dreamweaver, PHPedit, WebPage Editor,* dan lain sebagainya.

6) *Communication Application*

Aplikasi yang digunakan untuk membangun komunikasi antara satu komputer dengan komputer lain dalam jaringan komputer local maupun global. Contoh: *E-mail & Fax Application, Remote Access, Facebook chat, YM, GTalk, MSN, Terminal Emulation, Instant Messaging, Voice Recognition, Web Browser, Internet Telephony,* dan lain sebagainya.

7) Jenis Aplikasi Lain
   a) *Multimedia Player,* seperti *WinAmp, XMMS, Power DVD, JetAudio, Windows Media Player,* dan lain sebagainya.
   b) *Hacker Tools,* seperti *Find Password, Keygen/Crack Software License, Scanning Port,* dan lain sebagainya.
   c) *Network & Security Application,* seperti *Firewall, Intruder Detections, Network Monitoring,* dan lain sebagainya.
   d) *PC Games Apllication,* dan masih banyak lagi.

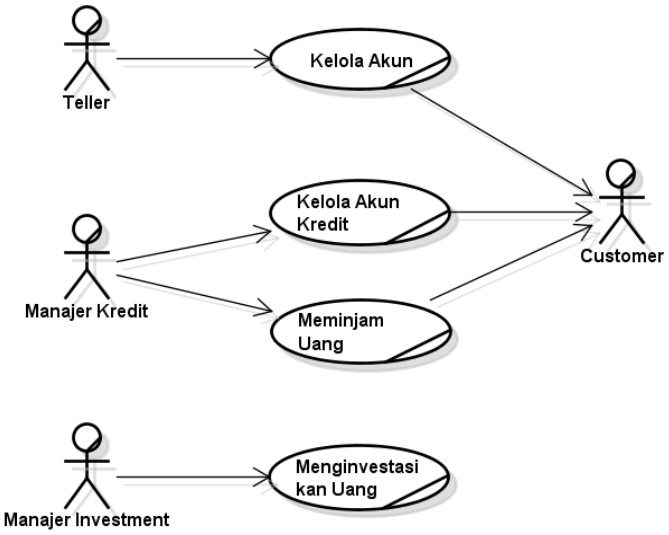
2.2.7 *Unified Modeling Language* (UML)

Unified Modeling Language (UML) adalah bahasa yang telah menjadi standar untuk visualisasi, pemodelan, perancangan dan mendokumentasikan artifak suatu sistem perangkat lunak berbasis objek yang terdiri dari kumpulan diagram, dikembangkan untuk membantu para pengembang sistem. Dikembangkan oleh Grady Booch, Ivar Jacobson dan James Rumbaugh dari Rational Software pada 1990-an. UML telah diadopsi oleh Object Management Group (OMG) pada tahun 2007 dan telah dikelola organisasi ini sejak saat itu. Pada tahun 2000 UML diterima oleh International Organization for

Standardization (ISO) sebagai standar untuk pemodelan sistem perangkat lunak intensif.

Hingga saat ini, UML telah berkembang hingga versi 2.5. UML memiliki banyak sekali diagram, terutama versi terbaru UML yaitu UML 2.5. Berikut akan dijelaskan macam-macam diagram yang paling sering digunakan dalam pembangunan aplikasi berbasis objek [9].

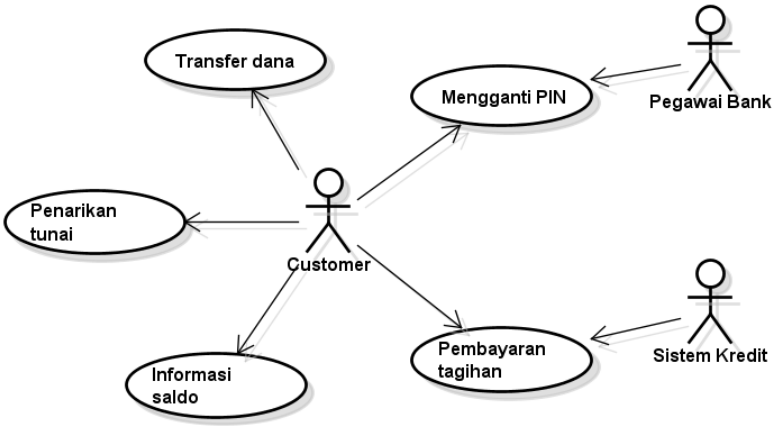
a. *Business Use Case* Diagram *Business use case* digunakan untuk mewakili fungsi yang disediakan oleh organisasi secara keseluruhan. *Business use case* juga digunakan secara luas selama kegiatan pemodelan bisnis untuk mengatur konteks sistem dan untuk membentuk sebuah fondasi untuk menciptakan use case.



Gambar 2. 2 Diagram Business Use Case untuk Institusi Finansial [9]

b. *Use Case* Diagram

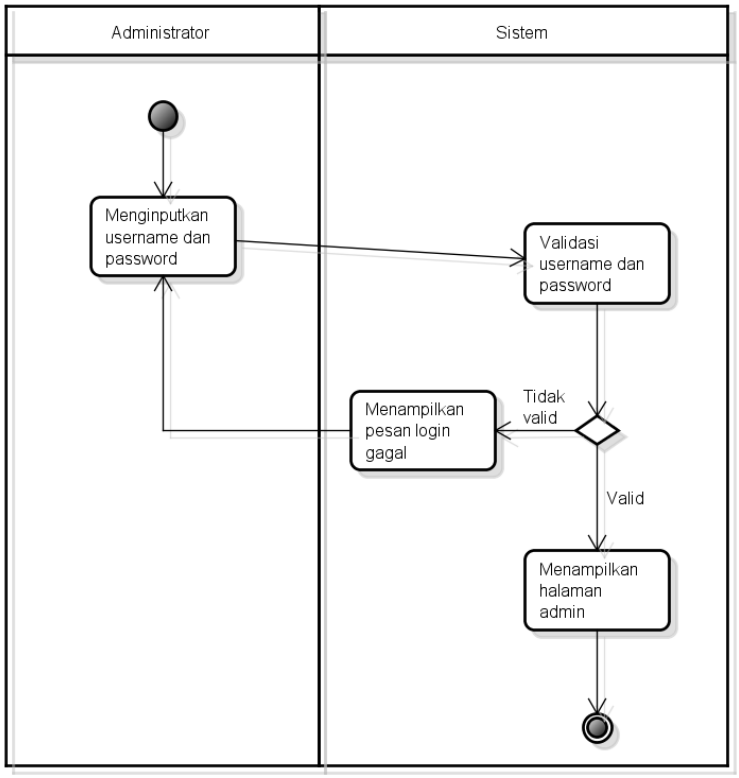
Use case digunakan untuk memodelkan proses bisnis berdasarkan perspektif pengguna sistem. *Use case* diagram terdiri atas diagram untuk use case dan actor. *Actor* merepresentasikan orang yang akan mengoperasikan atau orang yang berinteraksi dengan sistem aplikasi.



Gambar 2. 3 Diagram Use Case untuk Sistem ATM [9]

c. *Activity* Diagram

Activity diagram menggambarkan berbagai aliran aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing aliran berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity* diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity* diagram merupakan *state* diagram khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity* diagram tidak menggambarkan *behaviour* internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

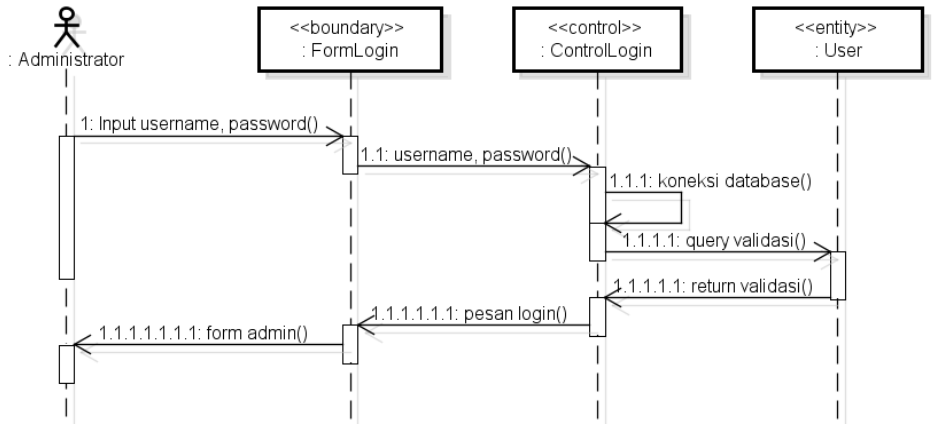


Gambar 2. 4 Diagram Aktifitas untuk Login Administrator [9]

d. *Sequence* Diagram

Sequence diagram menjelaskan secara detail urutan proses yang dilakukan dalam sistem untuk mencapai tujuan dari *use case*, diantaranya:

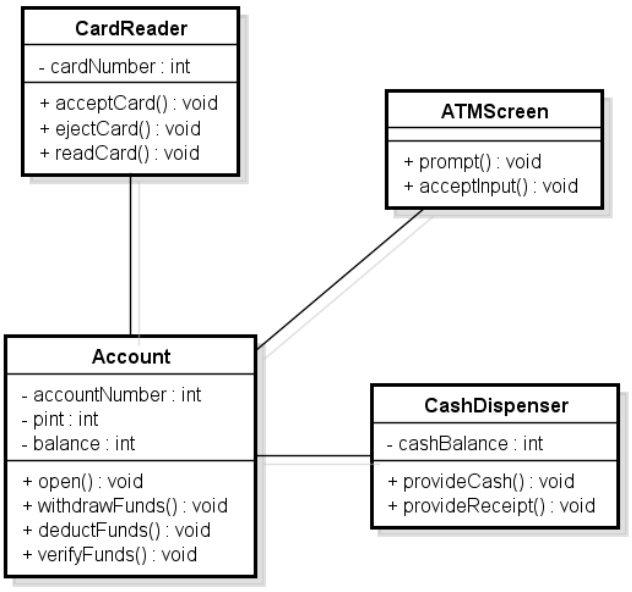
1) Interaksi yang terjadi antar *class*.
2) Operasi apa saja yang terlibat.
3) Urutan antar operasi.
4) Informasi yang diperlukan oleh masing-masing operasi.



Gambar 2. 5 Diagram Sequence untuk Proses Login [9]

e. *Class* Diagram

Class diagram merupakan diagram yang selalu ada di permodelan sistem berorientasi objek. *Class* diagram menunjukkan hubungan antar *class* dalam sistem yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai suatu tujuan. Selain itu, *class* diagram menunjukan juga atribut dan *method* yang ada pada setiap *class*.



Gambar 2. 6 Class Diagram untuk Use Case Penarikan Uang Sistem ATM [9]

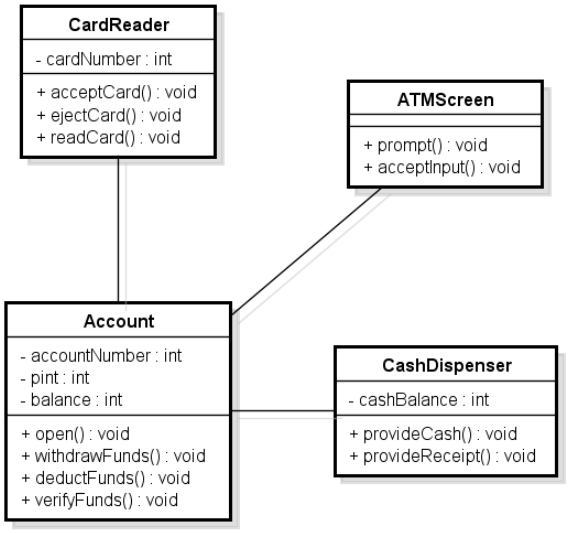
2.2.8 Object Oriented Programming (OOP)

Pemograman terstruktur adalah arus utama pada hari-hari sebelumnya dalam rekayasa perangkat lunak. Programmer mulai mengembangkan blok standar kode untuk melakukan operasi seperti pencetakan, dan kemudian menyalin dan menyisipkan kode tersebut ke setiap aplikasi yang mereka tulis [9]. Sementara hal ini mengurangi waktu pengembangan untuk aplikasi baru, cukup sulit jika perubahan yang diperlukan dalam blok kode, karena pengembang harus membuat perubahan di mana-mana. Dalam pemrograman terstruktur terdapat beberapa hal yang mana pemrograman berorientasi obyek dirancang untuk memecahkan hal tersebut.

Paradigma berorientasi objek adalah cara yang berbeda dalam memandang aplikasi. Pendekatan berorientasi objek, membagi aplikasi ke dalam banyak potongan kecil, atau benda, yang cukup independen satu sama lain. Kemudian aplikasi dibangun dengan menggabungkan semua benda atau potongan kecil ini bersama-sama. Beberapa prinsip orientasi objek diantaranya enkapsulasi, pewarisan, dan polimorfisme.

a. Enkapsulasi (*Encapsulation*)

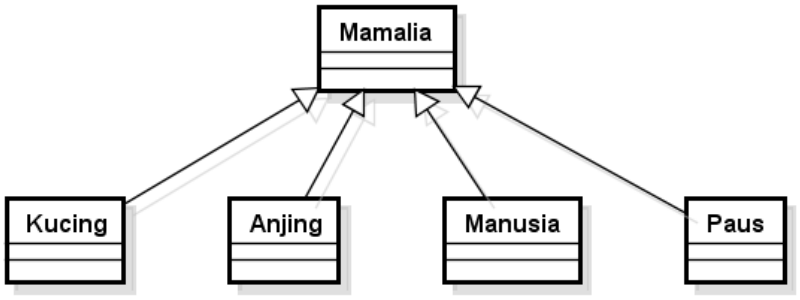
   Sistem berorientasi objek menggabungkan sepotong informasi dengan perilaku tertentu yang bertindak atas informasi tersebut. Lalu dirangkum menjadi sebuah objek. Hal ini disebut dengan enkapsulasi. Cara lain untuk melihat enkapsulasi adalah dengan membagi aplikasi menjadi bagian-bagian kecil dari fungsi terkait. Sebagai contoh, informasi yang berkaitan dengan rekening bank, seperti nomor rekening, saldo, nama pelanggan, alamat, jenis account, tingkat bunga, dan tanggal pembukaan. Lalu perilaku untuk rekening bank: membuka, menutup, deposit, withdraw, jenis perubahan, perubahan pelanggan, dan perubahan alamat. Setelah itu, informasi dan perilaku ini dirangkum menjadi sebuah objek akun. Akibatnya, setiap perubahan ke sistem perbankan mengenai rekening hanya dapat diimplementasikan dalam objek akun. Enkapsulasi bekerja seperti *one-stop shop* untuk semua informasi account dan perilaku.



Gambar 2. 7 Enkapsulasi Model Penarikan Uang di ATM [9]

b. Pewarisan (*Inheritance*)

Pewarisan adalah prinsip dasar berorientasi objek yang kedua. Dalam sistem berorientasi objek, pewarisan adalah mekanisme yang memungkinkan untuk membuat objek baru berdasarkan objek lama: Objek anak mewarisi sifat-sifat dari objek induk. Salah satu manfaat utama dari pewarisan adalah kemudahan dalam pemeliharaan. Ketika suatu perubahan yang mempengaruhi objek anak, hanya perlu mengubah objek orangtua maka objek anak secara otomatis akan mewarisi perubahannya. Contohnya jika mamalia tiba-tiba menjadi berdarah dingin, hanya objek mamalia akan perlu diubah. Kucing, anjing, manusia, ikan paus, dan benda-benda lainnya secara otomatis objek anak akan mewarisi karakteristik baru, yaitu karakteristik berdarah dingin dari mamalia .



Gambar 2. 8 Model Pewarisan Alami [9]

c. Polimorfisme (*Polymorphism*)

Prinsip ketiga orientasi objek adalah polimorfisme. Dari arti katanya, polimorfisme merupakan terjadinya berbagai bentuk, tahapan, atau jenis. Polimorfisme berarti memiliki banyak bentuk atau implementasi dari fungsi tertentu. Dalam hal sistem berorientasi objek, ini berarti bahwa perancang sistem dapat memliliki banyak implementasi dai fungsi tertentu seperti pada contoh kode untuk menggambar bangun ruang berikut.

```
Function Shape.drawMe()
{
SWITCH Shape.Type
Case "Circle"
   Shape.drawCircle();
Case "Rectangle"
   Shape.drawRectangle();
Case "Line"
   Shape.drawLine();
END SWITCH
}
```

Dengan polimorfisme, kode untuk menggambar hanya akan memanggil fungsi drawMe() seperti contoh kode berikut.

```
Function draw()
{
    Shape.drawMe();
}
```

Setiap bentuk (lingkaran, persegi panjang, dan garis) akan memiliki fungsi drawMe( ) untuk menggambar bentuk tertentu. Salah satu manfaat dari polimorfisme sama seperti prinsip-prinsip lain dari orientasi objek, yaitu kemudahan pemeliharaan.

2.2.9 MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (*database management system*) atau DBMS yang *multithread*, *multi-user*, dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis dibawah lisensi GNU *General Public License* (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL.

Berbeda dengan proyek-proyek seperti Apache, dimana perangkat lunak dikembangkan oleh komunitas umum, dan hak cipta untuk kode sumber dimiliki oleh penulisnya masing-masing, MySQL dimiliki dan disponsori oleh sebuah perusahaan komersial Swedia MySQL AB, dimana memegang hak cipta hampir atas semua kode sumbernya. Kedua orang Swedia dan satu orang Finlandia yang mendirikan MySQL AB adalah: David Axmark, Allan Larsson, dan Michael "Monty" Widenius. Perintah-perintah yang ada di SQL antara lain [6] :

a. Menambah data (*Insert*)

   Perintah SQL yang digunakan untuk menambah data ke *database* adalah perintah *INSERT*. Perintah ini mempunyai bentuk umum sebagai berikut:

```
INSERT [INTO] nama_tabel [(daftar field/kolom)]
VALUES (daftar_data).
```

b. Mengakses data (*Select*)

Perintah SQL yang digunakan untuk memilih data adalah perintah *SELECT*. Bentuk umum perintah ini adalah sebagai berikut:

```
SELECT daftar_select FROM daftar_tabel
[WHERE kondisi_pencarian]
[GROUP BY daftar_group_by]
[HAVING kondisi_pencarian]
[ORDER BY daftar_order[ASC|DESC ]]
```

c. Mengedit/Mengubah data (*Update*)

Perintah SQL yang digunakan untuk mengubah data adalah perintah *UPDATE SET*. Perintah ini mempunyai bentuk umum sebagai berikut:

```
UPDATE nama_table SET field1=databaru1
[,data2=databaru2]
[WHERE kondisi_update].
```

d. Menghapus data (*Delete*)

Perintah SQL yang digunakan untuk menghapus data adalah perintah *DELETE FROM*. Perintah ini mempunyai bentuk umum sebagai berikut:

```
DELETE FROM tabel_hapus WHERE kondisi_hapus.
```

MySQL memiliki beberapa kelebihan diantaranya [6] :

1. MySQL sebagai *Data Management System* (DBMS) dan *Relation Databese Management System* (RDBMS).
2. MySQL adalah *software database* yang *OpenSource*, artinya program ini bersifat *free* atau bebas digunakan oleh siapa saja tanpa harus membeli dan membayar lisensi kepada pembuatnya.
3. MySQL merupakan *database server*, jadi dengan menggunakan *database* ini dapat menghubungkan ke media *internet* sehingga dapat diakses jarak jauh.

4. MySQL merupakan sebuah *database client*. Selain menjadi *server* yang melayani permintaan, MySQL juga dapat melakukan *query* yang mengakses *database* pada *server*, jadi MySQL dapat juga berperan sebagai *Client*.

5. MySQL mampu menerima *query* yang bertumpuk dalam satu permintaan atau yang disebut *Multi-Threading*.

6. MySQL merupakan sebuah *database* yang mampu menyimpan data berkapasitas sangat besar hingga berukuran *Gigabyte* sekalipun.

7. MySQL didukung oleh driver ODBC, artinya *database* MySQL dapat diakses menggunakan aplikasi apa saja termasuk berupa visual seperti Delphi maupun Visual Basic.

8. MySQL adalah *database* menggunakan enkripsi *password*. Jadi *database* ini cukup aman karena memiliki *password* untuk mengaksesnya.

9. MySQL dapat menciptakan lebih dari 16 kunci per tabel dan dalam satu kunci memungkinkan berisi belasan *field*. Serta MySQL mendukung *field* yang dijadikan sebagai kunci *primer* dan kunci unik atau (*Unique*).
10. MySQL memiliki kecepatan dalam pembuatan tabel maupun peng-*update*-an tabel.

2.2.10 Sejarah Bahasa Pemrograman C#

Saat awal sistem operasi windows diperkenalkan oleh Microsoft Corp beberapa belas tahun lalu untuk menggantikan sistem operasi DOS (Disk Operating System), para programer pada umumnya menggunakan bahsa pemograman C untuk membuat program aplikasi yang akan dapat berjalan dengan baik diatas sistem operasi yang bersangkutan. Sebab utamanya adalah karena API (*Application Programing Interface*) *windows* yang banyak dimanfaatkan oleh program-program aplikasi itu juga ditulis menggunakan bahasa C sehingga kompatibilitasnya terjamin. Sementara hal itu dapat dilakukan dengan baik oleh para programer berbakat,untuk sebagian besar programer merasa kesulitan untuk penulisan bahasa pemograman C, sebab saat menulis programer harus melakukan

alokasi memori secara manual. Selain itu, kompleksitas bahasa pemograman C masih ditambah dengan adanya pointer-pointer aritmatika yang kompleks serta struktur program yang relatif sukar dibaca. Bahasa C sesungguhnya merupakan bahasa pemograman terstruktur, dimana kita tidak bisa memanfaatkan kelebihan bahasa pemograman yang memiliki paradigma Pemograman Berorientasi Objek (OOP – *Object Oriented Programing*) yang berkaitan dengan penggunaan ulang komponen (*component reusable*), seperti penyembunyian atau pembungkusan informasi (*encapsulation*), pewaris (*intheritance*), polimorfisme, dan sebagainya.

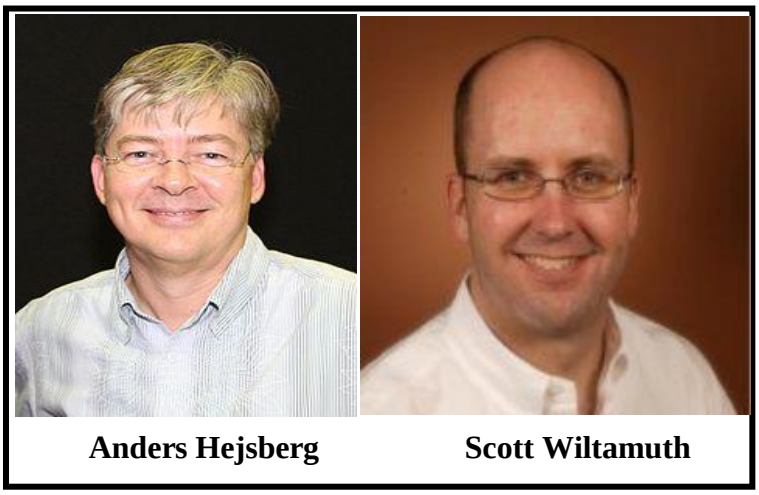
Selanjutnya, di masa-masa berikutnya, para programer bahasa C beralih ke bahasa pemograman C++ untuk melakukan pembangunan program di atas sistem operasi windows. Dalam hal ini bahasa pemograman C++ dapat dianggap sebagai perluasan dari bahasa pemograman C yang beradaptasi dengan konsep pemograman berorientasi objek. Bahasa pemograman C++ ini sama rumitnya dibanding bahasa pemograman C, meski memiliki keunggulan dalam hal adaptasinya dengan paradigma pemrograman berorientasi objek. Meski sangat rumi, beberapa *framework* C++ masih hadir saat ini, misalnya MFC (*Microsoft Foundation Classes*) yang menyediakan kelas-kelas yang dapat digunkan untuk mengembangkan aplikasi *Win32* API (*Application Programming Interface*).

Dalam hal di atas, *Microsoft Corp* cukup tanggap dalam menghadapi keluhan para programer akan kesulitan bahasa pemograman C/C++, yaitu dengan mengembangkan bahasa pemograman Visual Basic yang sangat populer dalam hal kemampuannya dan kemudahannya dalam mengembangkan antarmuka pengguna yang kompleks. Selain itu, *Microsoft Corp* juga menyediakan pustaka-pustaka kelas yang lengkap dalam bentuk COM (*Compounent Object Module*), serta logika akses basis data sangat mudah, bagi para pemrogram Visual Basic. Di atas MFC (*Microsoft Foundation Classes*), Visual Basic menyembunyika kompleksitas Win32 API dari pandangan pemrogram. Meski demikian, versi terakhir bahasa pemograman Visual Basic (VB 6) tidak secara sempurna beradaptasi dengan konsep pemograman berorientasi objek (OOP) sehingga mengurangi fleksibilitas dan pengunaan ulang komponennya.

Sementara itu, di sudut dunia pemograman yang lain, Sun Microsystem Inc mengembangkan bahasa pemograman Java yang memiliki sintaks cukup mirip dengan bahasa C/C++. Dalam hal ini, meskipun bahasa pemograman Java merupakan bahasa pemograman yang elegan dalam adaptasinya dengan pemograman berorientasi objek, integrasi antar program yang dikembangkan menggunakan bahasa pemograman java cukup sukar dilakukan karena sifatnya yang *open source,* yang mana setiap vendor di seluruh dunia memiliki varian produk yang kadang tidak kompatibel satu sama lain. Sebagai contoh, untuk mengimplementasikan konsep ORM (*Object Relational Mapping*), berbagai vendor di seluruh dunia mengembangkan kelas Hibernate, TopLink, iBaTIS, yang saling tidak kompatibel satu sam lain. Ini lain dengan produk *Microsoft Corp* juga mengembangkan objek – objek COM (*Component Object Module*) sedemikian rupa sehingga bisa diakses oleh pemrogram C/C++, Delphi, dan sebagainya, sehingga integrasi antar aplikasi dapat berlangsung dengan mulus. Meski demikian, COM (*Component Object Module*) juga memiliki kelemahan. Dalam hal ini, integraqsi antar aplikasi sepenuhnya berlangsung mulus pada aplikasi Internet yang di dalamnya terkandung berbagai teknologi yang beragam. Dari kenyatan ini, muncullah *framework* .NET yang berusaha sedemikian rupa sehingga integrasi antar aplikasi ini dapat berjalan dengan lebih mulus dan lebih sempurna lagi [9].

2.2.10.1 Bahasa Pemrograman C# Dan Platform .NET

Bahasa pemrograman C# (baca: C-Sharp) dirancang oleh Microsoft Corp. sebagai bahasa pemrograman yang sangat berdaya guna, aman (*secure*), serta mudah digunakan. Sebagai bagian dari platform .NET, bahasa pemrograman C# dirancang sedemikian rupa untuk bekerja dengan sangat baik di atas framework .NET yang mampu digunakan untuk menulis perangkat lunak handal demi layanan yang cepat. Bahasa pemrograman C# juga dapat digunakan untuk mengembangkan aplikasi sarana bergerak (*mobile application*), aplikasi berbasis Web (*Web-based application*), serta aplikasi berskala besar (*enterprise*).



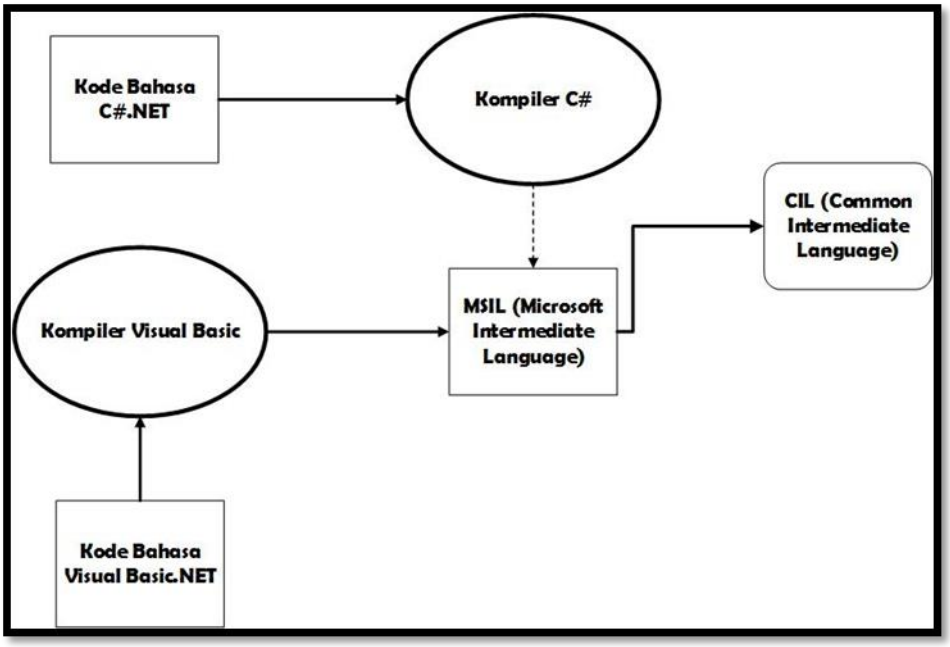
Gambar 2. 9 Para Penulis Sfesifikasi Bahasa C#

Selanjutnya mungkin ada pertanyaan mengapa kita membahas dan menggunakan bahasa pemrograman C# dalam buku ini ? penulis memilih bahasa pemrograman C# yang diimplementasikan oleh Microsoft Corp. dalam bentuk IDE (*Integrated Development Environment*) Visual C# sebagai basis penulisan buku ini karena bahasa pemograman C#, yang sesungguhnya merupakan salah satu dari bahasa keluarga C (C,C++, *Objective-C*, Java, dan sebagainya) terkenal dengan kecanggihannya, namun secara pemrograman mirip dengan bahasa pemrograman Visual Basic, yang terkenal dengan kemudahannya. Bahasa pemrograman C# merupakan gabungan dari kecanggihan bahasa keluarga Cdengan kemudahan pemrograman bahasa pemrograman Visual Basic.

Dalam hal di atas, spesifikasi awal bahasa pemrograman C# pada awalnya ditulis oleh Anders Hejsberg dan Scott Wiltamuth dari Microsoft Corp. dengan menggunakan bahsa pemrograman C# kita dapat mengembangkan logika aplikasi yang rmit seperti yang dapat dilakukan menggunakan bahasa keluarga C, namun juga dapat membuat antar muka pengguna yang ‘ramah’ terhadapa pengguna aplikasi (*user friendly*), serta dapat mengakses sistem basis data relasional (RDBMS – *Relational Database Management System*) dengan cara yang relatif mudah. Hal inilah yang membuat bahasa C# suatu saat akan menjadi salah satu bahasa pemrograman yang paling banyak digunakan di seluruh dunia.

Selanjutnya kita akan mengenal lebih jauh hal-hal yang berkaitan dengan bahasa pemrograman C#. Secara umum bahasa pemrograman C# memiliki karakteristik seperti yang tertulis di bawah ini :

1. Tidak ada alokasi memori secara manual menggunakan pointer
2. Manajemen memori otomatis menggunakan salah satu fiturnya yang dinamakan *garbage collection.*
3. Mendukung kontruksi kelas (*class*), antarmuka (*interface*), struktur, dan enumerasi seperti bahasa-bahasa pemograman berorientasi objek lainnya.
4. Mendukung pemrograman berbasis atribut (*ttribute-based programing*).
5. Mendukung LINQ (*Language Integrated Query*) yang memungkinkan aplikasi yang ditulis menggunakan bahasa pemrograman C# mampu berinteraksi dan bekerja bersama berbagai jenis format data di mana hal ini sangat penting saat kita membuat aplikasi bahasa C#yang mengakses sistem basis data relasional (RDBMS).
6. Mendukung tipe data dan kelas generik
7. Mendukung operator delegasi



Gambar 2. 10 Konversi Kode C# ke Microsoft Intermediate Language [10]

Saat ini kita melakukan kompilasi terhadap program yang kita tulis menggunakan bahasa pemograman C#, kompiler bahasa pemograman C# secara otomatis akan menerjemahkan kode-kode C# yang kita tulis menjadi suatu bentuk kode antara, yan disebut sebagai MSIL (*Microsoft Intermediate Language*), yang sama seperti kita menulis kode menggunakan Visual Basic .NET, Visual C++, dan sebagainya, sebelum kemudian menerjemahkan kembali menjadi berkas-berkas berekstensi .exe atau .dll di atas sistem operasi. Inilah alasan mengapa kita bisa menulis program dalam C#, Visual Basic, sedemikian rupa sehingga mereka masing-msing dapat saling memanfaatkan. *Multifle assembly* dalam hal ini merupakan proses kompilasi beberapa berkas .NET yang masing-masing dinamakan module dimana didalamnya juga berisi berkas *assembly manifest* yang memuat intruksi-intruksi CIL (*Common Intermediate Language*) dan metadata-metadata terkait.

Lalu komponen apa yang memungkinkan kode-kode bahasa pemograman C# dikompilasi agar kompatibilitas antar bahasa keluar .NET tetap terjaga ? Komponen yang dimaksud adalah *Just In Time Compiler* (JITers) yang bertindak

sedemikian rupa sehingga saat MSIL *(Microsoft Intermediate Language*) akan dieksekusi, CLR (*Common Language Runtime*) akan memanggil JITers yang akan menerjemahkan kode-kode bahasa keluarga .NET menjadi kode-kode yang spesifik terhadap komputer tertentu. JITers sesungguhnya peduli terhadap komputer-komputer tertentu yang akan digunakan untuk menjalankan program serta sistem operasi yang digunakannya. Dalam hal ini Microsoft Corp berani menjamin bahwa kode-kode bahasa dari platform .NET akan di eksekusi dengan cara yang lebih cepat dibanding dengan kode-kode yang ditulis menggunakan bahasa C/C++ tradisional.

Tidak seperti kode bahasa C/C++ tradisional, kode yang ditulis menggunakan bahasa pemrograman C# tidak secara langsung diterjemahkan ke bahsa mesin/komputer. Kompiler bahasa pemograman C# mengonversi kode yang ditulis menggunakan bahasa C# menjadi berkas MSIL, yang dinamakan *asembly*. Berkas MSIL dapat dibuat menggunakan semua bahasa pemrograman yang dikembangkan di atas CLR (*Common Language Runtime*), misalnya Visual C++, Visual Basic, serta Visual J#. Sekali lagi inilah alasa mengapa kita bisa menulis program dalam C#, Visual Basic, dan sebagainya, sedemikian rupa sehingga mereka masing-masing dapat memanfaatkan [10].

2.2.11 JSON

JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemprograman JavaScript, Standar ECMA-262 Edisi ke-3 - Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data.

JSON terbuat dari dua struktur:

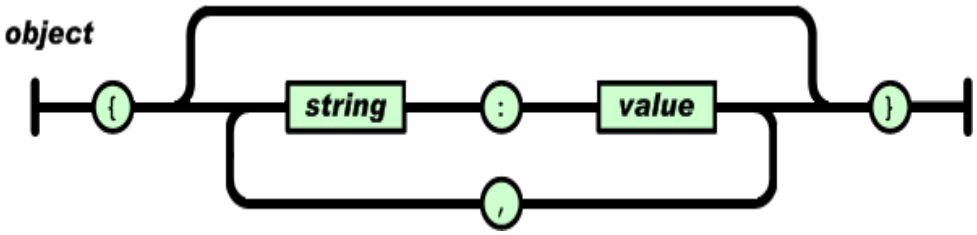
- Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus

(*dictionary*), tabel hash (*hash table*), daftar berkunci (*keyed list*), atau *associative array*.

- Daftar nilai terurutkan (*an ordered list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*sequence*).

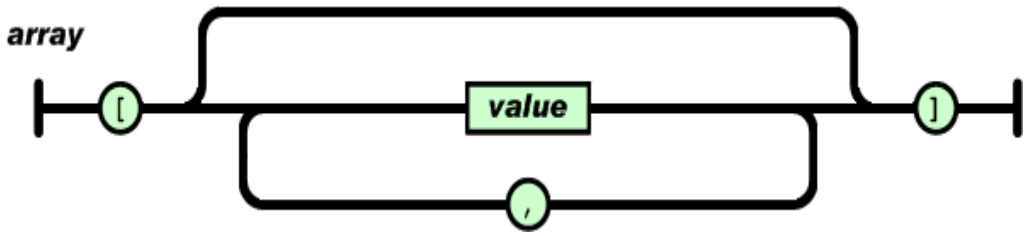
Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemprograman moderen mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemprograman yang juga berdasarkan pada struktur data ini. JSON menggunakan bentuk sebagai berikut [11]:

1. Objek adalah sepasang nama/nilai yang tidak terurutkan. Objek dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurung kurawal tutup). Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma).



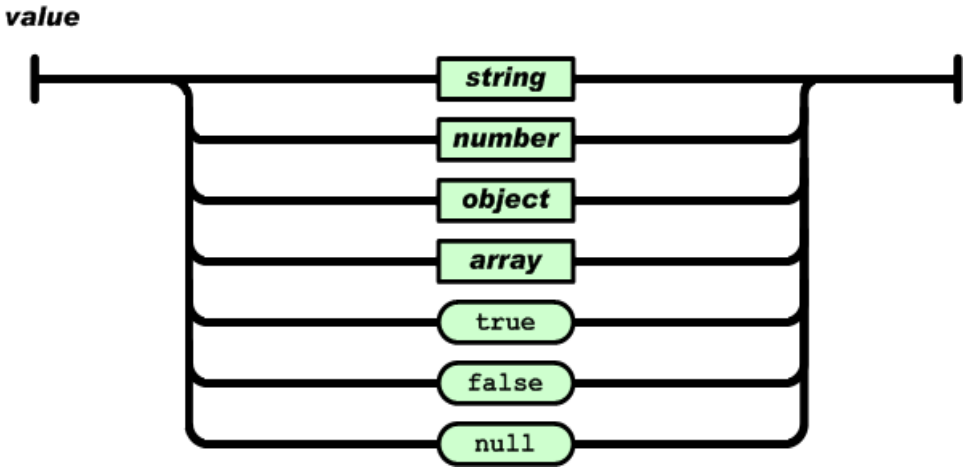
Gambar 2. 11 Penulisan Objek Pada C# [11]

2. Array adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [(kurung kotak buka) dan diakhiri dengan] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma).



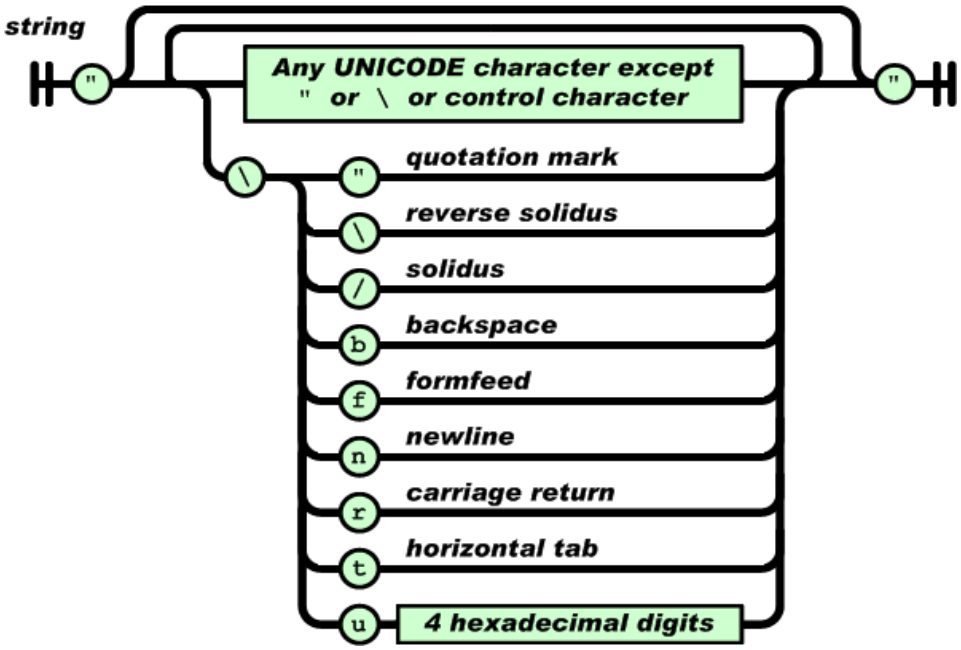
Gambar 2. 12 Penulisan Array Pada C# [11]

3. Nilai (*value*) dapat berupa sebuah string dalam tanda kutip ganda, atau *angka*, atau true atau false atau null, atau sebuah *objek* atau sebuah *larik*. Struktur-struktur tersebut dapat disusun bertingkat.



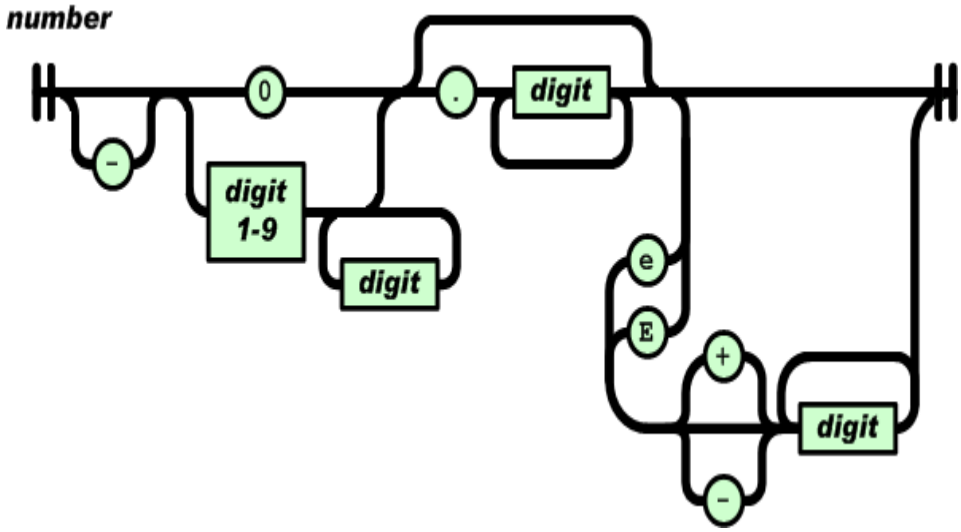
Gambar 2. 13 Nilai Pada C# [11]

4. String adalah kumpulan dari nol atau lebih karakter Unicode, yang dibungkus dengan tanda kutip ganda. Di dalam string dapat digunakan *backslash escapes* "\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada string. String sangat mirip dengan string C atau Java.



Gambar 2. 14 Format String Pada C# [11]

5. Angka adalah sangat mirip dengan angka di C atau Java, kecuali format oktal dan heksadesimal tidak digunakan.



Gambar 2. 15 Format Angka Pada C# [11]

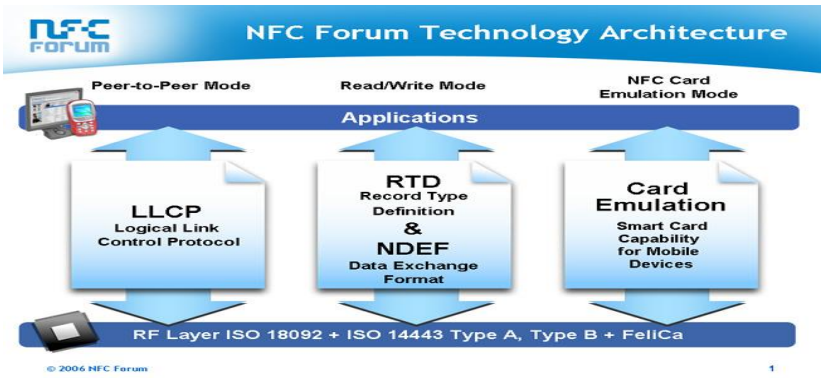
Spasi kosong (*whitespace*) dapat disisipkan di antara pasangan tanda-tanda tersebut, kecuali beberapa detil *encoding* yang secara lengkap dipaparkan oleh bahasa pemprograman yang bersangkutan.

2.2.12 Near Field Communication (NFC)

Near Field Communication (NFC) merupakan teknologi komunikasi baru dengan menggunakan induksi magnet berbasis teknologi *Radio Frequency Identification* (RFID). NFC mulai dikembangkan pada tahun 2002 oleh NXP dan Sony. Mereka berhasil membangun Forum NFC pertama bersama dengan Nokia. Sejak saat itu Forum NFC memiliki lebih dari 150 anggota dan peneliti yang terlibat dari beberapa perusahaan dan organisasi. Pada forum ini juga NFC dipromosikan dan secara resmi mengeluarkan sertifikasi untuk perangkat-perangkat yang mengadopsi teknologi NFC. NFC bisa juga disebut sebagai generasi kedua dari teknologi RFID.

NFC beroperasi pada frekuensi 13,56 MHz dengan kecepatan transmisi pengiriman mencapai 424 kbit/s. Jarak transmisi NFC sekitar 4-10 cm. Perbedaan antara NFC dan teknologi komunikasi *contactless* lainnya yaitu perangkat NFC dapat bersifat aktif – aktif (*peer to peer*) dan aktif – pasif. Oleh karena itu NFC

selalu melibatkan inisiator (*reader*) dan target. Inisiator aktif menghasilkan medan RF (*Radio Frequency*) yang dapat memberikan kekuatan ke target yang pasif (tidak memiliki sumber daya). Hal ini memungkinkan target NFC untuk memiliki bentuk yang sangat sederhana seperti stiker, gantungan kunci, atau kartu yang tidak memerlukan energy khusus [12].



Gambar 2. 16 Arsitektur NFC [12]

Banyak manfaat kegiatan-kegiatan yang diperoleh dengan memanfaatkan teknologi NFC. Seiring perkembangan jaman, invoasi-inovasi terhadap pemanfaatan NFC ini juga ikut berkembang. Perkembangan inovasi-inovasi tersebut terbagi ke dalam 3 kategori:

1. *Service Initiation*

Pada kategori ini NFC berguna untuk melakukan layanan sebagai media untuk mendapatkan informasi tertentu atau dapat juga untuk membuka layanan yang mengharuskan autentifikasi data terlebih dahulu. Hal ini melibatkan NFC *Tag* sebagai media penyimpanan maupun penyampaian sementara. Pada *Tag* tersebut dapat berisi informasi teks, alamat sebuah *website* (*url),* nomor telepon atau data lain yang ukuran yang kecil.

Salah satu contoh penggunaannya yaitu pada poster pintar (*smart poster*). Dengan mendekatkan ponsel ada kepada NFC *Tag* pada poster, maka pengguna akan langsung mendapatkan informasi lebih detail pada yang belum tertera pada poster tersebut tanpa harus menginputkan apapun pada ponselnya.

2. *Peer-to-peer*

Pada kategori ini NFC berperan dalam mengkomunikasikan antara dua perangkat aktif dengan ukuran data yang relatif kecil. Namun untuk kebutuhan data yang berukuran besar, NFC dapat digunaka sebagai sambungan nirkabel seperti *Bluetooth* dan WiFi.

Contoh dari penggunaannya yaitu pada proses pencetakan foto langsung dari ponsel atau kamera. Pengguna tinggal memilih foto yang akan dicetak, kemudia mendekatakn ponselnya kepada printer yang sudah terintegrasi dengan teknologi NFC untuk membuat sambungan *bluetooth* untuk mengirimkan gambar dan mencetaknya.

3. *Payment*

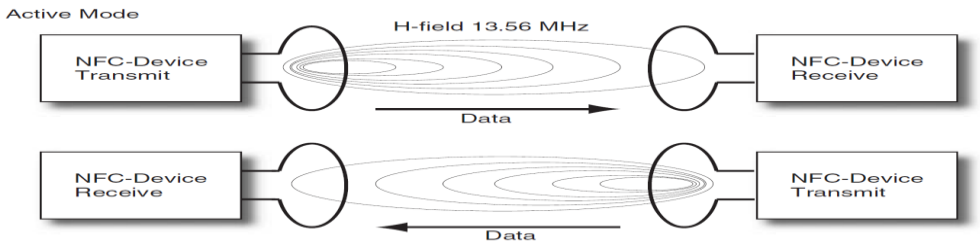
Pada kategori ini NFC berperan dalam proses pembayaran suatu barang yang sudah terintegrasi sebelumnya dengan kartu kredit, bank dan operator jaringan lainnya. Dengan memanfaatkan teknologi NFC dalam pembayaran ini dapat mempermudah dan mempercepat proses transaksi pembayaran karena tidak menggunakan uang tunai maupun metode pembayaran konvensional lainnya.

Contoh dari penggunaanya yaitu pada proses transaksi pembayaran di restoran, pusat perbelanjaan dan area parkir.

Komunikasi NFC dapat dilakukan dalam 2 mode yaitu mode aktif dan mode pasif.

1. Mode Aktif

Pada mode aktif, NFC yang berperan sebagai initiator mengaktifkan transmiter yang dimiliki sehingga arus dengan frekuensi tinggi yang mengalir pada antena menimbulkan medan magnetik H yang menyebar ke sekitarnya seperti terlihat pada Gambar 2.17.



Gambar 2. 17 Mode Aktif [12]

Sebagian akan bergerak dan tertangkap oleh antena interface NFC yang terletak pada lokasi berdekatan. Kemudian tegangan U dihasilkan di antena yang dapat dideteksi dan diterima oleh receiver pada NFC target.

Cara kerja tersebut mirip dengan konsep RFID Reader dan transponder. Namun, terdapat perbedaan dimana pada transponder RFID reader harus memberikan suplai energi kepada transponder untuk mengoperasikan microchip untuk komunikasi data. Sementara pada NFC, perangkat elektronik yang memiliki fitur NFC yang memberikan suplai energi.

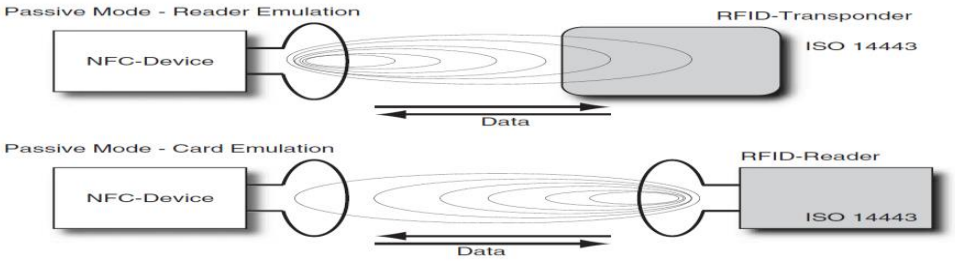
Pada mode aktif, pengiriman data dilakukan dari NFC target kepada NFC initiator dengan NFC target mengaktifkan transmitter dan NFC initiator berpindah mengaktifkan receiver. Sehingga pada mode ini kedua interface NFC bergantian menghasilkan medan magnetik.

2. Mode Pasif

Pada mode pasif, NFC inisiator juga menghasilkan medan magnetik untuk memulai komunikasi dengan NFC target. Namun setelah paket data sampai ke target, NFC inisiator tidak mengganti mode menjadi penerima seperti NFC mode aktif, melainkan tetap mengaktifkan transmitter. Sehingga trasmitter hanya diaktifkan pada NFC inisiator.

Kemudian data pada NFC target akan "ditumpangkan" pada gelombang tersebut dengan menghasilkan yang disebut sebagai *load modulation*. Hal ini memberikan keuntungan dalam pengoperasiannya pada kondisi daya yang dimiliki oleh NFC target sedang lemah.

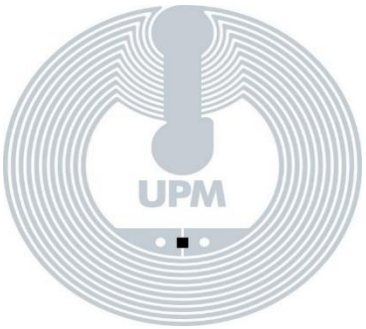
Hal ini juga memungkinkan *interface* NFC berfungsi seperti RFID *Reader* untuk berkomunikasi dengan transponder pasif dengan memberikan suplai daya pada transponder dan menggunakan *load modulation* data ditransimisikan pada interface NFC. Misalnya, *smartphone* dengan fitur NFC dapat digunakan untuk membaca dan menulis pada transponder seperti *smart label* atau *e-ticket*. Aktivitas *interface* NFC yang menyerupai RFID *Reader* disebut juga '*reader-emulation mode*'.



Gambar 2. 18 Mode Pasif [12]

2.2.13 NFC *Tag*

NFC *Tag* merupakan media pendukung teknologi NFC. NFC Tag biasanya bersifat pasif, sehingga membutuhkan inisiator (NFC bersifat aktif) untuk proses komunikasi. NFC *Tag* dapat berisi data dengan ukuran kecil. . NFC *Tag* saat ini memiliki memory berkisar antara 96 dan 4.096 byte. NFC *Tag* berisfat *read-only* tetapi memungkinkan juga bersifat *re-writeable* [13].



Gambar 2. 19 NFC Tag [13]

Produsen NFC telah menentukan standar dari NFC *Tag* untuk dapat dioperasikan dengan perangkat NFC. Spesifikasi yang ditentukan terbagi menjadi 4 tipe. Keempat tipe *Tag* NFC tersebut didasarkan pada produk dan ketersediaan komersil. Berikut adalah 4 tipe dari NFC *Tag* :

1. Tipe 1 : didasari pada ISO/IEC 14443A. *Tags* mampu membaca dan menulis ulang. Pengguna dapat mengatur tag menjadi *read-only*. Memori yang tersedia adalah 96 bytes dan dapat di-*upgrade* hingga 2 Kbyte.
2. Tipe 2 : didasari pada ISO/IEC 14443A. *Tags* mampu membaca dan menulis ulang. Pengguna dapat mengatur tag menjadi *read-only*. Memori yang tersedia adalah 48 bytes dan dapat di upgrade hingga 2 Kbyte.
3. Tipe 3 : didasari pada standar industri Jepang atau *Japanese industrial standard* (JIS) X 6319-4, atau disebut Felica. *Tag* diatur pada pembuatannya untuk mendukung baik dalam *read* dan *re-writeable*, ataupun *read-only*. Memori yang tersedia dapat diubah, secara teori batas memorinya adalah 1 Mbyte per *service*.
4. Tipe 4 : kompatibel sepenuhnya dengan ISO/IEC 14443 standard *series*, memiliki memori 32 KB dan lebih cepat daripada tipe *tag* lainnya, memiliki proteksi terhadap data *collision*, dapat menggunakan komunikasi NFC-A maupun NFC-B. Tipe 4 dapat ditulis ulang *(rewriteabl*e) atau *read-only*. Tetapi pengaturan tersebut dilakukan saat tipe 4 dalam tahap pembuatan. Saat sudah selesai dibuat, pengaturannya tidak dapat diubah [13].

2.2.14 NFC Reader

NFC reader adalah perangkat yang menerima atau mengirimkan perintah dari NFC. NFC reader akan sangat berguna bagi pedagang yang ingin menyediakan layanan pembayaran menggunakan NFC [14].



Gambar 2. 20 Reader NFC (Near Field Communication) [14]

2.2.15 Tabel URI (Uniform Resource Identifier)

Untuk mempersingkat URI, byte pertama dari data record menggambarkan protokol dari URI. Tabel tersebut harus digunakan untuk proses encoding dan decoding URI, meskipun aplikasi mungkin menggunakan nilai 0x00 untuk menunjukkan tidak adanya awalan saat encoding, terlepas dari adanya kode singkatan yang sesuai. Untuk penjelasan dari protokol yang berbeda, silahkan lihat tabel URI berikut :

Tabel 2. 1 Format URI Forum NFC

Value	Protocol
0x00	URI *Field*
0x01	*http://www.*
0x02	*https://www.*
0x03	*http://*
0x04	*https://*
0x05	*Tel:*
0x06	*mailto:*

Value	Protocol
0x08	*ftp://ftp.*
0x09	*ftps://*
0x0A	*sftp://*
0x0B	*smb://*
0x0C	*nfs://*
0x0D	*ftp://*
0x0E	*dav://*
0x0F	*news:*
0x10	*telnet://*
0x11	*imap:*
0x12	*rtsp://*
0x13	*urn:*
0x14	*pop:*
0x15	*sip:*
0x16	*sips:*
0x17	*tftp:*
0x18	*btspp://*
0x19	*btl2cap://*
0x1A	*btgoep://*
0x1B	*tcpobex://*
0x1C	*irdaobex://*
0x1D	*file://*
0x1E	*urn:epc:id:*
0x1F	*urn:epc:tag:*
0x20	*urn:epc:pat:*
0x21	*urn:epc:raw:*
0x22	*urn:epc:*
0x23	*urn:nfc:*

Sebagai contoh, jika isi baris dari tabel URI adalah 0x02, dan datanya berisi "nfcforum.org", maka yang dihasilkan adalah"https://www.nfcforum.org". Jika isi baris dari tabel URI adalah nol (0x00), maka awalan akan dilakukan. Semua kolom yang ditandai RFU akan diperlakukan seolah-olah bernilai nol (tidak ada awalan). Sebuah sistem yang kompatibel tidak harus menghasilkan nilai-nilai yang ditandai dengan RFU.

2.2.15.1 URI *Field*

Kolom URI ini menyediakan URI sesuai RFC 3987 [RFC 3987] (yang sebenarnya merupakan IRI, atau internasionalisasi Resource Identifier, tetapi

sejak dahulu sudah ditetapkan dengan kata URI). IRI ini bisa menjadi URL atau URN seperti yang dijelaskan sebelumnya. Pengkodean yang digunakan harus UTF-8, kecuali skema URI menentukan beberapa encoding tertentu.

Panjang IRI dapat dihitung dengan mengambil panjang payload(muatan), dan mengurangkan 1 untuk protokol kode singkatan byte.Ini merupakan panjang dalam satuan byte, Ini merupakan panjang dalam satuan byte, tidak dalam karakter (seperti UTF-8 karakter yang dapat menempati lebih dari satu byte).

URI didefinisikan hanya dalam 7-bit ruang US-ASCII. Oleh karena itu, aplikasi yang kompatibel harus mengubah UTF-8 IRI string ke bentuk 7-bit US-ASCII string dengan mengubah poin kode di atas 127 ke pengkodean yang tepat. Kode ini telah didefinisikan dalam RFC 3987 [RFC 3987] dan IDN [RFC 3492] dokumen. Untuk skema yang berbeda, pengkodean juga mungkin berbeda.
Misalnya, jika URI (setelah awalan dari jenis bidang URI) mengandung string berikut: "Http: //www.hääyö.com/", itu diubah, sesuai standar IDN [RFC 3492] aturan, menjadi "http://www.xn--hy-viaa5g.com" sebelum melakukan proses selanjutnya. Kebanyakan aplikasi modern yang sudah mendukung skema internasionalisasi Resource Identifier (IRI) yg baru. Hal ini direkomendasikan untuk implementasi yang mendukung IRI dimana tampilan dari form URI dapat dibaca manusia dengan mudah.

Untuk memperjelas, URI mungkin mengandung karakter UTF-8. Namun, Internet tidak bisa menangani kaarakter tersebut, dan oleh karena itu URI perlu diubah sebelum digunakan. Untuk sebagian besar perangkat, konversi ini dilakukan oleh aplikasi.

Setiap nilai karakter dalam URI antara (dan termasuk) 0 dan 31 akan dicatat sebagai kesalahan (error), dan *record* URI akan dibuang. Setiap data yang tidak sesuai dengan urutan UTF-8 akan dianggap kesalahan, dan seluruh catatan URI akan dibuang. 4 Handling Guideline / 4 Pedoman Penanganan URI RTD tidak mendefinisikan aksi khusus jika perangkat sesuai untuk melakukan proses tersebut. Ini merupakan implementasi dari aksi yang akan dilakukan. Berikut contoh tentang cara menggunakan URI RTD dalam aplikasi :

1. Menyimpan URL Sederhana

   Untuk menempatkan URL http://www.nfc.com pada tag menggunakan protokol NDEF, tambahkan urutan byte berikut. Total panjang: 12 bytes.

Tabel 2. 2 Menyimpan Data URL

Offset	Content	Explanation
0	0xD1	SR = 1, TNF = 0x01 (NFC Forum Well Known Type), ME=1, MB=1
1	0x01	Length of the Record Type (1 byte)
2	0x08	Length of the payload (8 bytes)
3	0x55	The URI record type (“U”)
4	0x01	URI identifier (“http://www.”)
5	0x6e 0x66 0x63 0x2e 0x63 0x6f 0x6d	The string “nfc.com” in UTF-8.

2. Menyimpan No Telepon

   Untuk menyimpan nomor telepon (misalnya, untuk membuat ponsel melakukan panggilan ke nomor ini), menggunakan urutan byte berikut. Nomor itu adalah '+ 358-9-1234567'. Total panjang adalah 17 byte.

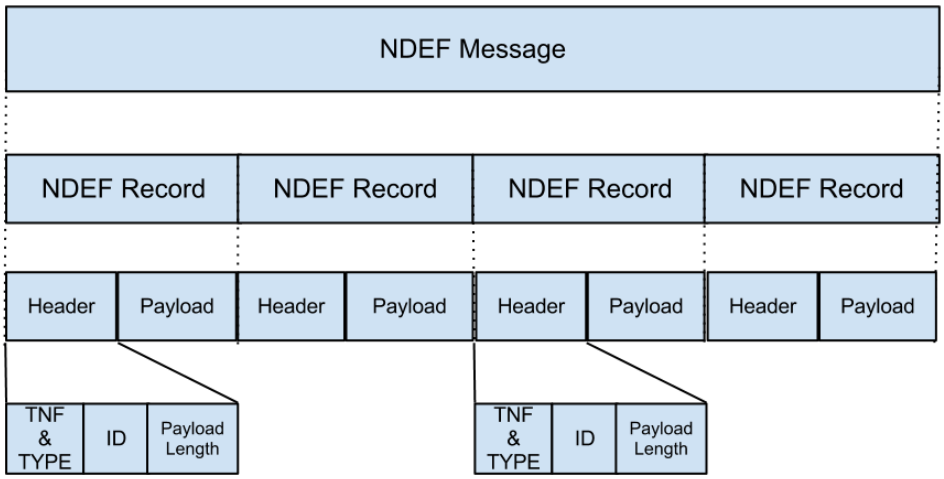
Tabel 2. 3 Menyimpan Data No Telepon

Offset	Content	Explanation
0	0xD1	SR = 1, TNF = 0x01 (NFC Forum Well Known Type), ME=1, MB=1
1	0x01	Length of the Record Type (1 byte)

Offset	Content	Explanation
2	0x0D	Length of the payload (13 bytes)
3	0x55	The URI record Name (“U”)
4	0x05	URI identifier (“tel:”)
5	0x2b 0x33 0x35 0x38 0x39 0x31 0x32 0x33 0x34 0x35 0x36 0x37	The string “+35891234567” in UTF-8.

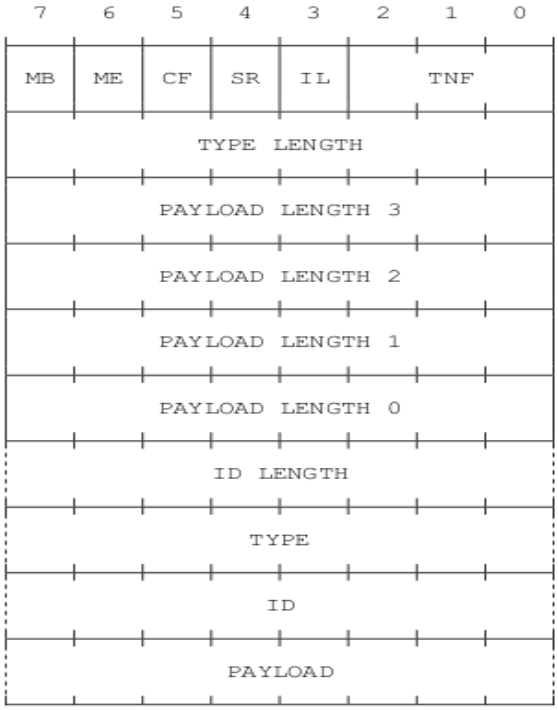
2.2.16 NFC *Data Exchange Format* (NDEF)

NFC Data Exchange Format (NDEF) adalah format pesan standar pada proses pertukaran informasi antara perangkat NFC ke perangkat NFC dan perangkat NFC ke *Tag* NFC. Di dalam NDEF terdapat NDEF *Message* yang terdiri dari beberapa *record* seperti pada gambar 2.21 [13].



Gambar 2. 21 *Struktur NDEF* Message *[13]*

NDEF *Record* merupakan format umum dari panjang variabel *record.* Berikut adalah struktur NDEF *Record.*



Gambar 2. 22 Struktur NDEF Record [13]

Berikut adalah keterangan dari struktur NDEF *Record* :

a. MB (*Message Begin*) : Area 1 bit. Ketika diset maka akan menunjukan awal NDEF *Message*.

b. ME (*Message End*) : Area 1 bit. Ketika diset maka akan menunjukan akhir NDEF *Message*.

c. CF (*Chunk Flag*) : Area 1 bit. Untuk mengidentifikasi apakah potongan *record* pertama atau potongan tengah *record*.

d. SR (Short Record) : Area 1 bit. Jika diset maka PAYLOAD_LENGTH merupakan oktet tunggal. Short Record ini dimaksudkan untuk

menenkapsulasi *Payload*, dengan kisaran ukuran 0 sampai 255

e. IL (*Identification Length*) : Area 1 bit. Jika diset maka ID_LENGTH akan muncul di bagian *header* sebagai oktet tunggal. Jika ID_LENGTH bernilai 0, ID_LENGTH dihilangkan dari *header record* dan area ID dihilangkan dari *record.*

f. TNF (*Type Name Format*) : Area bernilai 3 bit. Untuk mengidentifikasi struktur dari *value* TYPE.

Tabel 2. 4 Value Type Name Format

Type Name Format	***Value***
Empty	0x00
NFC Forum well-known type [NFC RTD]	0x01
Media-type as defined in RFC 2046 [RFC 2046]	0x02
Absolute URI as defined in RFC 3986 [RFC 3986]	0x03
NFC Forum external type [NFC RTD]	0x04
Unkown	0x05
Unchanged	0x06
Reserved	0x07

Keterangan tabel 2.1 *Value Type Name Format* :

a. 0x00 (Empty)

Menunjukan bahwa tidak ada jenis atau payload yang terkait dengan *record*. Ketika digunakan TYPE_LENGTH, ID_LENGTH dan bidang PAYLOAD_LENGTH harus bernilai nol dan TYPE_ID dan PAYLOAD dihilangkan dari *record*. Nilai TNF ini dapat digunakan setiap kali *record* kosong diperlukan, misalnya untuk mengakhiri NDEF *message* ketika tidak ada PAYLOAD yang diinputkan oleh pengguna aplikasi.

b. 0x01 (NFC Forum wel-known type [NFC RTD])

Menunjukan bahwa TYPE berisi nilai yang mengikuti *Type Name Format* RTD sesuai spesifikasi pada NFC Forum.

c. 0x02 (Absolute URI as defined in RFC 3986 [RFC 3986])
   Menunjukan bahwa TYPE berisi nilai yang sesuai dengan yang dibuat oleh BNF dan didefinisikan dalam RFC 2046.
d. 0x03 (Absolute URI as defined in RFC 3986 [RFC 3986])
   Menunjukan bahwa TYPE berisi nilai yang mutlak sesuai dengan URI BNF dan didefinisikan dalam RFC 3986.
e. 0x04 (NFC Forum external type [NFC RTD])
   Menunjukkan bahwa bidang TYPE berisi nilai yang mengikuti *Type Name Format* RTD yang didefinisikan dalam NFC Forum RTD spesifikasi [NFC RTD] untuk nama jenis eksternal.
f. 0x05 (Unkown)
   Menunjukan jika tipe PAYLOAD tidak diketahui. Hal ini serupa dengan "application/octet-stream" jenis media yang didefinisikan oleh MIME RFC 2046. Ketika digunakan TYPE_LENGTH harus bernilai nol. Maka dengan demikian TYPE dihilangkan dari NDEF *record*.
g. 0x06 (Unchanged)
   Hanya untuk *record* tengah dan mengakhiri *record* dari Chunked Payload. TYPE_LENGTH harus bernilai nol.
h. 0x07 (Reserved)
   Milik NFC Forum untuk digunakan di masa depan.

g. TYPE_LENGTH
   TYPE_LENGTH adalah 8 bit integer yang menentukan panjang oktet dari TYPE.
h. ID_LENGTH
   ID_LENGTH adalah 8 bit integer yang menentukan panjang oktet dalam ID. ID_LENGTH muncul apabilan IL diset menjadi 1 di dalam *record header*. ID_LENGTH boleh bernilai 0 oktet namun ID harus dihilangkan dari NDEF *record*.

i. PAYLOAD_LENGTH

PAYLOAD_LENGTH adalah yang menentukan panjang oktet dari PAYLOAD.

j. TYPE

Nilai TYPE adalah identifikator yang menjelaskan jenis nilai PAYLOAD. Nilai dari TYPE harus sesuai dengan struktur, encoding dan format pada TNF.

k. ID

Nilai dari ID adalah identifier dalam bentuk resferensi URI yang dijelaskan dalam RFC 3986. Referensi URI dapat bersifat relative dan absolut karena NDEF tidak mendefiniskan basis URI.

l. PAYLOAD

Konten yang dapat berisi apa saja dalam ukuran beberapa bytes.