# BAB II
# TEORI PENUNJANG

## I.1 Bencana Banjir Menurut BPDB

Banjir merupakan peristiwa ketika air menggenangi suatu wilayah yang biasanya tidak digenangi air dalam jangka waktu tertentu. Banjir biasanya terjadi karena curah hujan turun terus menerus dan mengakibatkan meluapnya air sungai, danau, laut atau drainase karena jumlah air yang melebihi daya tampung media penopang air dari curah hujan tadi.

Selain disebabkan faktor alami, yaitu curah hujan yang tinggi, banjir juga terjadi karena ulah manusia. Contoh, berkurangnya kawasan resapan air karena alih fungsi lahan, penggundulan hutan yang meningkatkan erosi dan mendangkalkan sungai, serta perilaku tidak bertanggung jawab seperti membuang sampah di sungai dan mendirikan hunian di bantaran sungai.

Kejadian bencana banjir sangat bersifat lokal. Satu daerah bisa terlanda banjir dan daerah lainnya aman. Oleh sebab itu, informasi mengenai banjir yang resmi biasanya berasal dari institusi di daerah yang bertanggung jawab, seperti BPBD.

Kendati sifatnya bencana lokal, namun terkadang banjir juga dapat meluas dan melumpuhkan kehidupan perkotaan seperti yang pernah terjadi di Jakarta. Oleh sebab itu, langkah antisipasi harus dilakukan baik sebelum, saat, dan pascabencana banjir [4].

*Gambar 2.1.1.1* Bencana Banjir

### I.1.1 Status Bencana Banjir

1. Siaga IV

Belum ada peningkatan debit air secara mencolok. Komando di lapangan, termasuk membuka atau menutup pintu air serta akan dikemanakan arah air cukup dilakukan oleh komandan pelaksana dinas atau wakil komandan operasional wilayah [4].

2. Siaga III

Bila hujan yang terjadi menyebabkan terjadinya debit air meningkat di pintu - pintu air tetapi kondisinya masih belum kritis dan membahayakan. Meski demikian, bila status siaga III sudah ditetapkan, masyarakat sebaiknya mulai berhati-hati dan mempersiapkan segala sesuatunya dari berbagai kemungkinan bencana banjir [4].

3. Siaga II

Bila hujan yang terjadi menyebabkan debit air mulai meluas, maka akan ditetapkan Siaga II, penanggung jawab untuk siaga II ini adalah Kepala Badan Penanggulangan Bencana Daerah Prov. DKI Jakarta yaitu Sekretaris Daerah [4].

4. Siaga I

Bila dalam enam jam debit air tersebut tidak surut dan kritis maka ditetapkan Siaga I. Penanggung jawab penanganan status siaga I langsung di tangan Gubernur [4].

## I.2 Mitigasi Banjir

Menurut (Ciottone, 2006), mitigasi adalah segala sesuatu yang meliputi jenisyang luas dari perhitungan yang dilakukan sebelum suatu kejadian terjadi yangmana akan mencegah korban sakit, cidera, dan meninggal serta mengurangi sekecil-kecilnya dampak kehilangan harta benda. Rencana mitigasi pada umumnya meliputi kemampuan untuk memelihara fungsi, desain bangunan, lokasi bangunandi luar dari zona bahaya, kemampuan esensial bangunan, proteksi dari bagian darisuatu bangunan, asuransi, edukasi publik, peringatan, dan evakuasi. Mitigasi dilaksanakan sebelum, sesudah, dan sebelum terjadinya suatu bencana.Untuk bencana banjir sendiri, salah satu tindakan mitigasi bencana banjir adalahmelakukan peringatan dini bencana banjir. Salah satu contoh apabila tidak adaperingatan dini banjir, maka semua daerah yang dilalui aliran banjir akan memakankerugian yang besar. Pada daerah hulu, dapat dilakukan beberapa cara peringatandini, seperti: menempatkan pengukur hujan di hulu dengan akses komunikasi kewilayah hilirnya, melakukan identifikasi jenis material yang terbawa arus banjir, dan melihat dan mengamati kondisi awan dan lamanya hujan (Paimin, 2009). Sesuai dengan Peraturan Menteri Dalam Negeri No. 33 Tahun

2006 tentang Pedoman umum mitigasi bencana menjelaskan tentang langkah-langkah

yang dilakukan dalam mitigasi bencana banjir seperti: pengawasan penggunaan lahan, pembangunan infrastruktur yang kedap air, pengerukan dan pembangunan sudetansungai, pembuatan tembok pemecah ombak, pembersihan sedimen, pembuatansaluran drainase, pelatihan pertanian yang sesuai dengan daerah banjir, dan juga enyiapkan persiapan evakuasi bencana banjir [5].

## I.3 B4A

*Basic4Android* merupakan sebuah tool RAD *(Rapid Aplication Development)* yang digunakan untuk membuat aplikasi berbasis *Android,* dimana *Android* adalah sebuah sistem operasi untuk *smartphone* atau table yang sedang berkembang pesat dan begitu popular saat ini [6].


Gambar 2.1.1.2 B4A

## I.4 MySQL

MySQL adalah sebuah server database open source yang terkenal yang digunakan di berbagai aplikasi terutama untuk server atau membuat WEB. MySQL adalah sebuah implementasi dari sistem manajemen basisdata relasional (RDBMS) yang didistribusikan secara gratis dibawah lisensi GPL (*General Public License*). Setiap pengguna dapat secara bebas menggunakan MySQL, namun dengan batasan perangkat lunak tersebut tidak boleh dijadikan produk turunan yang bersifat komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam basisdata yang telah ada sebelumnya.

SQL adalah sebuah konsep pengoperasian basisdata, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis.

Kehandalan suatu sistem basisdata (DBMS) dapat diketahui dari cara kerja pengoptimasi-nya dalam melakukan proses perintah-perintah SQL yang dibuat oleh pengguna maupun program-program aplikasi yang memanfaatkannya. Sebagai peladen basis data, MySQL mendukung operasi basisdata transaksional maupun operasi basisdata non-transaksional. Pada modus operasi non-transaksional, MySQL dapat dikatakan unggul dalam hal unjuk kerja dibandingkan perangkat lunak peladen basisdata kompetitor lainnya. Namun demikian pada modus non-transaksional tidak ada jaminan atas reliabilitas terhadap data yang tersimpan, karenanya modus non-transaksional hanya cocok untuk jenis aplikasi yang tidak [7].


Gambar 2.1.1.3 *Mysql*

## I.5 *Android*

*Android* adalah sebuah sistem operasi untuk perangkat *mobile* berbasis *linux* yang mencakup sistem operasi, *middleware*, dan aplikasi. *Android* adalah sistem operasi untuk telepon seluler yang berbasis *Linux*. *Android* menyediakan *platform* terbuka bagi para pengembang untuk membuat aplikasi mereka sendiri.

Pada awalnya dikembangkan oleh *Android Inc*, sebuah perusahaan pendatang baru yang membuat perangkat lunak untuk ponsel yang kemudian di beli oleh Google Inc. Untuk pengembangannya, di bentuklah *Open Handset Alliance* (OHA), konsorsium dari 34 perusahaan perangkat keras, perangkat lunak, dan telekomunikasi termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia [8].

### I.5.1 Versi *Android*

**a. *Andoid* versi 4.1 (*Jelly Bean*)**

*Android* versi 4.1 di rilis dengan penambahan fitur baru, diantaranya dukungan terhadap *OpenGL ES 3.0* yang menjadikan performansi tinggi pada sektor grafis, selain itu terdapat juga fitur bluetooth smart yang dapat menghemat daya pada saat pemakaian Bluetooth [8].

**b. *Andoid* versi 4.4 (*KitKat*)**

*Android* versi 4.4 di rilis pada tanggal 31 Oktober 2013 dengan pembaharuan antarmuka, optimasi kinerja pada *perangkat* dengan spesifikasi rendah, peningkatan tampilan mode layar penuh, dan dukunagn Bluetooth Message Access Profile (MAP) [8].

**c. *Android* versi 5.0 (*Lollipop*)**

*Android* versi 5.0 di rilis pada tanggal 3 November 2014 dengan perbaharuan antarmuka dengan warna yang lebih hidup. Terdapat beberapa fitur baru, diantaranya penghemat baterai, device sharing, notifikasi, desain materia, dan keamanan yang lebih baik [8].

**d. *Android* versi 6.0 (*Marshmallow*)**

*Android* 6.0 di rilis pada tahun 2015. Tujuan marshmallow memoles sudut kasar dan membuat versi lollipop lebih baik lagi. fitur baru dari versi ini yaitu dukungan sidik jari resmi untuk perangkat, dukungan untuk pembayaran seluler melalui *Android* pay, model perizinan yang lebih baik untuk aplikasi, google now di tap dan deep menghubungkan apps [8].

**a. *Android* versi 7.0 (*Nougat*)**

*Android* 7.0 di rilis pada Tahun 2016. Fitur baru dari versi ini adalah Doze on the Go untuk waktu siaga yang lebih baik lagi, multi window untuk penggunaan dua aplikasi secara bersamaan, aplikasi setelan yang lebih baik, hapus semua di layar aplikasi baru-baru ini, balas langsung ke pemberitahuan, notifikasi di bundel, pengaturan cepat akan mengubah kustomisasi [8].

**b. *Android* versi 8.0 (*Oreo*)**

*Android* 8.0 di rilis pada Maret 2017.Fitur baru dari versi ini adalah pemberitahuan untuk prioritas dan kategorisasi yang lebih baik, pengelolaan warna lebih baik, *Android* o memiliki koleksi emoji baru yang telah di desain ulang, waktu boot lebih cepat: pada perangkat pixel, sekarang bisa mengalami waktu boot dua kali lebih cepat dibandingkan dengan nougat, mengisi otomatis dan mengingat kata sandi dalam aplikasi [8].

## I.6 UML

*Unified Modelling Language* (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa pemrograman berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian, UML tetap dapat digunakan untuk *modeling* aplikasi prosedural dalam VB atau C [9].

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax* / semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna

tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*). Sejarah UML sendiri cukup panjang. Sampai era tahun 1990 seperti kita ketahui puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs- brock, dsb. Masa itu terkenal dengan masa perang metodologi (*method war*) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila kita bekerjasama dengan group/perusahaan lain yang menggunakan metodologi yang berlainan [9].

Di mulai pada bulan Oktober 1994 Booch, Rumbaugh dan Jacobson, yang merupakan tiga tokoh yang boleh di kata metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 direlease draft pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut di koordinasikan oleh *Object Management Group* (OMG). Tahun 1997 UML versi 1.1 muncul, dan saat ini versi terbaru adalah versi 1.5 yang di rilis bulan Maret 2003. Booch, Rumbaugh dan Jacobson menyusun tiga buku serial tentang UML pada tahun 1999. Sejak saat itulah UML telah menjelma menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek [9].

### I.6.1 Use Case Diagram

*Use case* diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah use case merepresentasikan sebuah interaksi antara aktor dengan sistem. Use case merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, membuat sebuah daftar belanja, dan sebagainya. Seorang/sebuah actor

adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu [9].

*Use case* diagram dapat sangat membantu bila kita sedang menyusun requirement sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang test case untuk semua feature yang ada pada sistem. Sebuah use case dapat menambahkan fungsionalitas use case lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa use case yang ditambahkan akan dipanggil setiap kali use case yang menambahkan dieksekusi secara normal [9].

Sebuah use case dapat ditambahkan oleh lebih dari satu use case lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang common. Sebuah use case juga dapat melakukan extend use case lain dengan behaviour yang dimiliki. Sementara hubungan generalisasi antar use case menunjukkan bahwa use case yang satu merupakan spesialisasi dari yang lain [9].

### I.6.2 Class Diagram

*Class* diagram adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut / properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi) [9].

Class diagram menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. *Class* memiliki tiga area pokok :

a. Nama

b. Atribut

c. Metode

Atribut dan metoda dapat memiliki salah satu sifat berikut :

a. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan.
b. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.

a. *Public*, dapat dipanggil oleh siapa saja.

*Class* dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class*. Dengan demikian *interface* mendukung resolusi metoda pada saat *run-time*. Sesuai dengan perkembangan *class* model, *class* dapat dikelompokkan menjadi *package*. Kita juga dapat membuat diagram yang terdiri atas *package*.

Hubungan Antar *Class* :

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class*.
2. Agregasi, yaitu hubungan yang menyatakan bagian ("terdiri atas"). Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
3. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence* diagram yang akan dijelaskan kemudian.

### I.6.3 Activity Diagram

*Activity* Diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity* diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi [9].

*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.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan *behaviour* pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal. *Activity* diagram dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

### I.6.4 Sequence Diagram

*Sequence* diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence* diagram terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait) [9].

*Sequence* diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang menjadi *trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan. Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal.

Message digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan di petakan menjadi operasi/metoda dari *class*. *Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya di awali dengan diterimanya sebuah *message*. Untuk objek- objek yang memiliki

sifat khusus, standar UML mendefinisikan icon khusus untuk objek *boundary*, *controller* dan *persistent entity*.

## I.7 PHP

*Hypertext Preprocessor* atau di singkat PHP adalah suatu bahasa pemerograman yang digunakan untuk membuat web dinamis, walau bisa juga digunakan untuk membuat program lain. Tentunya bahasa pemerograman PHP berbeda dengan HTML, pada PHP Script/kode yang di buat tidak dapat di tampilkan pada halaman/muka website begitu saja, tapi harus diproses terlebih dahulu oleh web server lalu di tampilkan dalam bentuk halaman website di web browser, Script PHP juga dapat di sisipkan pada HTML dan script PHP selalu diawali dengan <?php dan di akhiri dengan ?>. Manajamen database yang biasanya digunakan untuk pemerograman PHP misalnya seperti MySQL, tapi ada juga yang menggunakan Oracle, Microsoft Access, dan lain-lain. PHP disebut juga sebagai bahasa pemerograman script server side, karena PHP di proses pada komputer server [10].

## I.8 Metode *Bayesian Network*

*Bayesian Network*s merupakan suatu metode pemodelan data berbasis probabilitas yang merepresentasikan suatu himpuan variabel dan conditional interdependenciesnya melalui suatu DAG (Directed Acyclic Graph). roses yang dilakukan yaitu parameter learning dan structure learning. Structure learning umumnya dilakukan dengan proses try and error, dengan model dari semua node ke model hasil atau dari satu node ke model hasil. Umumnya metode yang banyak digunakan adalah metode pertama, dimana semua node dan kaitan antar mereka dimunculkan dan kemudian menguji semua kaitan serta menghilangkan kaitan yang tidak mempunyai nilai keterkaitan yang tinggi [3].