BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Penelitian terdahulu bertujuan untuk mendapatkan bahan perbandingan dan acuan. Selain itu, untuk menghindari anggapan kesamaan dengan penelitian ini. Maka dalam kajian pustaka ini peneliti mencantumkan hasil-hasil penelitian terdahulu sebagai berikut:

Tabel 2.1 Penelitian Terdahulu

Nama	Judul	Variabel	Metode Analisis	Hasl Analisis
Aji Dedi Mulawarman, Novrida Qudsi Lutfillah, Febrina Nur Ramadhani, Meryana Ananda (2022)	Aplikasi Sistem Informasi Bank Sampah: Sebuah Studi Kasus	Sistem Informasi Bank Sampah	Action Research	Sistem informasi tersebut didesain dengan maksud selain untuk mempermudah operasional bank sampah, juga untuk pengendalian, efisiensi, dan meminimalisasi komplain ketidaksesuaian pencatatan antara nasabah dengan petugas atau praktisi
S Atin, S Mutia, A Widayanti, H S Yatawa, A A Rafdhi, I Afrianto (2022)	Perancangan Sistem Informasi Bank Sampah Berbasis Website	Sistem Informasi Bank Sampah	Pendekatan UCD	Hasil dari penelitian ini menunjukkan bahwa perancangan website yang dibangun dapat memudahkan operator dalam mengelola data sampah dan tabungan warga, dapat tersimpan dalam basis data dan dapat diakses setiap saat. Disamping itu bagi warga, sistem ini memiliki manfaat untuk mempercepat proses pengambilan sampah di lingkungannya, serta mendapatkan keuntungan dari penjualan sampah tersebut menjadi tabungan

2.2 Bank Sampah

Bank Sampah merupakan konsep pengumpulan sampah kering dan dipilah serta memiliki manajemen layaknya perbankan tapi yang ditabung bukan uang melainkan sampah. Warga yang menabung yang juga disebut nasabah memiliki buku tabungan dan dapat meminjam uang yang nantinya dikembalikan dengan sampah seharga uang yang dipinjam (Atin *et al.*, 2022).

Bank sampah merupakan salah satu cara pengelolaan sampah skala rumah tangga, yang menitik beratkan pada pemberdayaan masyarakat dalam mengelola sampah rumah tangga. Bank sampah adalah tempat menabung sampah yang telah terpilih menurut jenis sampah, sampah yang ditabung pada bank sampah adalah sampah yang mempunyai nilai ekonomis (Yustanti, 2017).

Bank Sampah adalah termpat perbankan yang aktivitas nya sama dengan perbankan pada umumnya, sampah-sampah anorganik yang terdapat pada sektor rumah tangga, sekolah, tempat umum dapat memiliki nilai ekonomi apabila masyarakat dapat memilah sampah tersebut dengan baik. Tidak semua jenis sampah anorganik yang dapat dibeli oleh bank sampah, ada beberapa jenis sampah yang dapat dibeli seperti alumunium, botol, kertas, duplex dan lain-lain (Evan, *et al.*, 2021). Menurut Undang-Undang RI Nomor 10 tahun 1998 tanggal 10 November 1998 tentang perbankan, yang dimaksud dengan bank adalah badan usaha yang menghimpun dana dari masyarakat dalam bentuk simpanan dan menyalurkan kepada masyarakat dalam bentuk kredit dan atau bentuk-bentuk lainnya dalam rangka meningkatkan taraf hidup rakyat banyak (Pamungkas *et al.*, 2020).

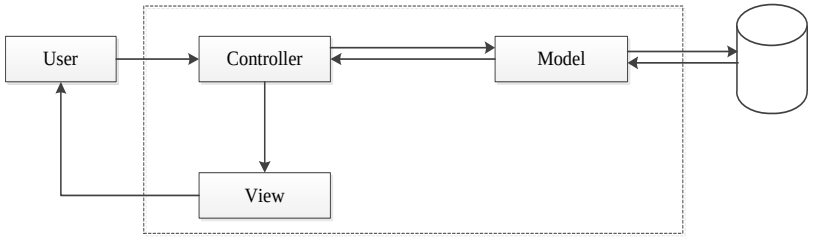
2.3 *CodeIgniter*

CodeIgniter adalah *Framework* untuk bahasa pemprograman PHP, yang dibuat Rick Ellis pada tahun 2006. *CodeIgniter* memiliki banyak fitur yang membantu para pengembang PHP untuk dapat membuat aplikasi secara mudah dan cepat serta memiliki sifat yang fleksibel dapat mengembangkan dalam perangkat *web*, *dekstop* maupun *mobile* (Raharjo, 2018).

Codeigniter adalah sebuah aplikasi gratis yang berupa kerangka kerja untuk membangun website menggunakan bahasa pemprograman PHP (Heru, 2018).

Framework merupakan kumpulan dari fungsi atau prosedur dan class untuk tujuan tertentu yang sudah siap untuk digunakan untuk membuat program tanpa harus membuat fungsi atau class dari awal. *Framework codeIgniter* dipilih untuk meningkatkan kualitas dari pengembangan dan kode sistematisas (Widaningsih dan Suheri, 2019).

CodeIgniter memilki konsep atau pola *Model-View-Controller* (MVC) sehingga kode-kode dapat di sederhanakan.



Gambar 2.1 Arsitektur MVC

2.3.1 *Website*

Website merupakan halaman yang menampilkan informasi data teks, gambar, suara, video atau gabungan dari semuanya, baik yang bersifat statis maupun dinamis. Halaman pertama sebuah *website* disebut *homepage*. *Website*/Situs merupakan kumpulan informasi atau kumpulan halaman/*page* yang bisa diakses lewat jalur *internet*. Setiap orang di berbagai tempat dan segala waktu bisa menggunakannya selama terhubung secara *online* (Harianto, *et al.*, 2019).

World Wide Web (WWW) atau sering disebut Web merupakan salah satu sumber daya internet yang berkembang pesat. Informasi Web didistribusikan dengan pendekatan hyperlink yang memungkinkan seseorang memeperoleh informasi dengan meloncat dari satu halaman ke halaman lain (Juliany, *et al.*, 2018).

2.3.2 PHP

PHP (*Personal Home Page*) adalah pemrograman (interpreter) yang melakukan proses penerjemahan baris sumber menjadi kode mesin yang dimengerti oleh komputer secara dinamis. Pengertian PHP juga merupakan singkatan dari *Hypertex Preprocessor* dengan Bahasa yang berbentuk skrip yang

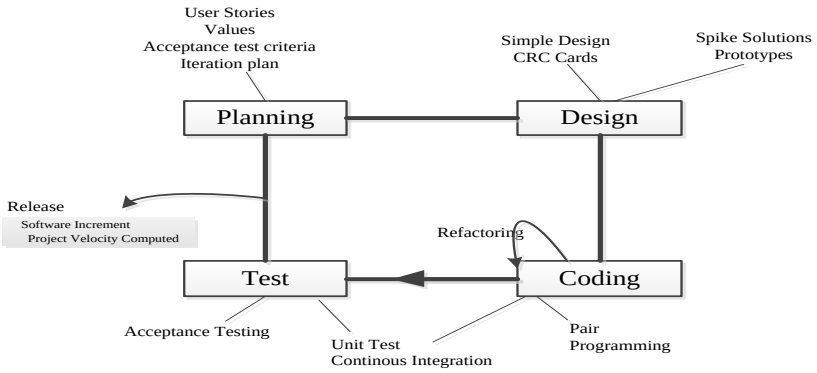
bersifat *server side* yang dimana proses pengerjaan kode program dilalukan di *server*, dan hasilnya akan ditampilkan di *browser* (Sihombing, 2022).

2.3.3 *MySql*

MySQL adalah sebuah *database management system* (manajemen basis data) menggunakan perintah dasar SQL (*Structured Query Language*) yang cukup terkenal. Database *management system* (DBMS) MySQL multi pengguna dan bersifat gratis. Mysql digunakan sebagai wadah dalam mengelola data yang dapat disimpan didigunakan kembali dengan cara yang lebih efesien (Setyawan and Pratiwi, 2019).

2.4 Metode *Extreme Programming*

Extreme programming berdasarkan sejarah singkat bahwa pengembangan perangkat lunak banyak digunakan untuk pengembangan yang lebih cepat dengan meliputi tahapan *planning, design, coding* dan *testing* . Berikut merupakan konsep *Extreme programming* (Suryantara, 2017).



Gambar 2.2 *Extreme Programming*

2.4.1 Sejarah Singkat *Extreme Programming*

Extreme Programming merupakan salah satu metodologi rekayasa perangkat lunak yang banyak digunakan untuk pengembangan aplikasi oleh para developer, diperkenalkan oleh kent beck yang ditunjukan untuk menangani sebuah proyek penggajian yang dikenal sebagai C3 *(Chrysler Comprehensive Compensation)*. Proyek tersebut dimulai pada maret 1996 yan terancam gagal karena rumitnya sistem mengalami kegagalan pada proses testing, kemudian

pihak Chrysler menyewa Kent Beck sebagai konsultan di bidang *software engineering* yang kemudian disebut sebagai pencetus XP.

2.4.2 Kerangka Kerja *Extreme Programming*

Pengembangan yang dilakukan menggunakan XP dengan proses yang lebih cepat dengan tahapan seperti *planning, design, coding* dan *testing.*

1. *Planning/*Perencanaan

   Tahap ini dimulain dengan pemahaman konteks bisnis dari aplikasi dengan mendefinisikan keluaran seperti fitur, fungsi, penentuan waktu dan biaya serta alur pengembangan.

2. *Design/*Perancangan

   Tahap perencanaan secara sederhaana dengan alat mendesain kartu CRC (*Class Responsibility Collaborator*) yang digunakan untuk pemetaan kelas-kelas yang akan digunakan pada diagram UML.

3. *Coding/*Pengkodean

   Hal utama dalam pengembangan menggunakan XP yaitu *pari programming* (Proses pembuatan program melibatkan 2 atau lebih programmer).

4. *Testing/*Pengujian

   Tahap ini fokus pada pengujian fitur pada aplikasi sehingga tidak ada kesalahan dan sesuai dengan proses bisnisnya.

2.4.3 Nilai Inti *Extreme Programming*

Nilai inti pada pengembangan sistem yaitu :

1. *Communication* (Komunikasi)

   Komunikasi antar tim yang digunakan untuk saling berbagi pengetahuan dalam pengembangan.

2. *Simplicity* (Kesederhanaan)

   Melakukan semua proses dengan sederhana dan mencoba mencari solusi yang paling sederhana.

3. *Feedback* (Masukan)

   Masukan untuk mengetahui kemajuan dari proses dan kualitas perangkat lunak yang dibuat.

4. *Courage* (Kesalahan)

   Kesuksesan pengembangan aplikasi harus memiliki keberanian, keyakinan dan integritas dalam pekerjaan.

5. *Respect* (Menghormati)

   Menerapkan siklus pendek dan integrasi *continue.*

2.4.4 Aspek Dasar *Extreme Programming*

Aspek dasar pada penerapan metode *extreme programming* yaitu :

1. *The Planning Game*

   Proses pendek dan cepat, mengutamakan aspek teknik, memisahkan unsur bisnis dengan unsur teknis dan pertemuan intensif antara klien dengan developer. Pada XP proses ini menggunakan terminologi “game” karena Beck menyarankan untuk menggunakan teknik *score card* dalam menentukan *requirements*.

2. *Small Releases*

   Menyelesaikan bagian - bagian aplikasi dan melakukan persentasi kepada *client*, setelah mendapatkan persetujuan maka dilkukan penerapan keaplikasi.

3. *Metaphor*

   Menggambarkan visi yang luas terhadap tujuan dari pengembangan perangkat lunak. Dengan Tujuan diharapkan komunikasi antara klien dengan developer akan berlangsung lebih baik.

4. *Simple Design*

   Menghindari desain yang rumit dalam sebuah pengembangan perangkat lunak. Dengan desain yang simpel apabila terjadi perubahan dapat meminimalkan kesalahan.

5. *Refactoring*

   Melakukan perubahan pada kode program dari perangkat lunak dengan tujuan meningkatkan kualitas dari struktur program tersebut tanpa mengubah cara program tersebut bekerja

6. *Testing*

   Membuat test terhadap aplikasi berdasarkan model test yang telah ditentukan.

7. *Pair Programming*

   Dua orang programer saling bekerjasama di komputer yang sama untuk menyelesaikan sebuah unit .

8. *Colletive Ownership*

   Saling berbagi pengetahuan agar tidak saling ketergantungan pada programer tertentu ataupun berbagai hambatan akibat perbedaan gaya menulis program dapat diperkecil.

9. *Coding Standard*

   Dengan adanya coding standard yang telah disepakati terlebih dahulu maka pemahaman terhadap program akan menjadi mudah untuk semua programer dalam tim.

10. *Continuous Integration*

    Melakukan *build* sesering mungkin berbagai kesalahan pada program dapat dideteksi dan diperbaiki secepat mungkin.

11. *40-hours Week*

    Beck berpendapat bekerja 8 jam sehari dan 5 hari seminggu adalah maksimal untuk tiap programer.

12. *On-Site Customer*

    XP menganjurkan bahwa ada anggota dari klien yang terlibat pada proses pengembangan perangkat lunak. Apabila ada kesalahan dalam pengembangan diharapkan klien dapat segera memberikan masukan untuk koreksinya.

2.4.5 Tujuan *Extreme Programming*

Tujuan metode *extreme programming* untuk menghasilkan perangkat lunak yang berkualitas tinggi dan lebih produktif dan mengurangi biaya selama ada perubahan dalam pengembangan perangkat unak menggunakan siklus pengembangan perangkat lunak singkat.

2.4.6 Daur Hidup Metodologi *Extreme Programming*

Metode XP dapat diterapkan bila:

1. Adanya perubahan yang sangat cepat
2. Memiliki resiko yang tinggi pada pembuatan aplikasi
3. Dalam tim pengembangan aplikasi dengan sedikit programmer
4. Mampu mengotomatisasikan uji sistem
5. Keterlibatan peran serta klien secara langsung
6. Harus ada komunikasi yang baik

2.4.7 Keuntungan dan Kerugian *Extreme Programming*

Keuntungan pada penerapan metode XP yaitu:

1. Dalam hal XP menjalin komunikasi yang baik dengan klien pada pengembangan aplikasi
2. Saling menghargai antar developer dan meningkatkan komunikasi
3. Dapat menjadi pembelajaran bagi orang lain
4. Klien mendapatkan umpan balik yang akurat mengenai aplikasi yang dibuat
5. Dengan XP dapat mengubah pemikiran pelanggan terhadap aplikasi yang dibuat
6. Developer tidak berkerja secara berlebihan
7. Dengan XP dapat membuat keputusan yang bersifat teknikal

2.5 *Unified Modelling Language (UML)*

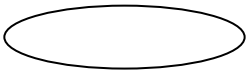
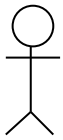
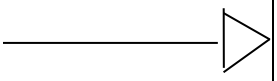
Alat pengembang sistem merupakan konsep desain yang digunakan untuk menggambarkan sistem dengan menggunakan diagram. Penyesuaian alat yang digunakan harus sesuai dengan metode pengembangan yang dilakukan salah satunya adalah penerapan *Unified Modelling Language.* Menurut (Rosa dan Salahuddin, 2019), *Unified Modelling Language* adalah bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung. Berikut ini merupakan penjelasan tentang masing-masing diagram yang ada pada *Unified Modelling Language.*

2.5.1 *Use Case Diagram*

Use Case adalah *use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. *Use case* digunakan untuk

mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut (Rosa and Shalahuddin, 2019). Berikut simbol-simbol yang akan digunakan dalam menggambarkan *Use Case Diagram* dapat dilihat pada Tabel 2.2.

Tabel 2.2 Simbol *Use Case Diagram*

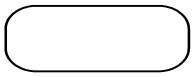
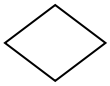
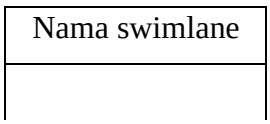
No	Simbol	Deskripsi
1.		*Use case*: Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal *frase* nama *use case*.
2.		Aktor: seseorang/sesuatu yang berinteraksi dengan yang akan dibuat. diluar sistem informasi. Biasanya dinyatakan menggunakan kata benda
3.		Asosiasi *(association)*: merupakan komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau use case memiliki interaksi dengan aktor.
4.		Generalisasi *(generalization):* merupakan hubungan (umum – khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum
5.	<< Include >>	Include berarti *use case* yang ditambahkan akan dipanggil saat use case tambahan dijalankan.
6.	<<Extend>>	Ekstensi *(extend)* merupakan use case tambahan ke sebuah *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu.

2.5.2 *Activity Diagram*

Activity diagram adalah *activity* diagram menggambarkan aliran kerja atau aktivitas dari sebuah sistem atau proses bisnis atau menggambarkan aktivitas

sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem (Rosa dan Salahuddin, 2019). Berikut simbol-simbol yang akan digunakan dalam menggambarkan *activity diagram* dapat dilihat pada Tabel 2.3.

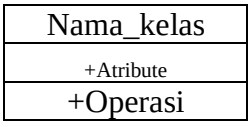
Tabel 2.3 Simbol *Activity Diagram*

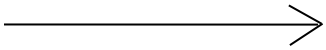
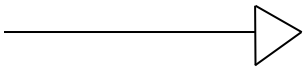
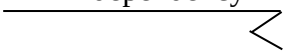
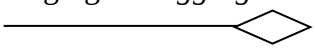
No.	Simbol	Keterangan
1.		Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
2.		Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
3.		Percabangan *(Decision)* merupakan asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
4.		Penggabungan *(Join)* merupakan asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
5.		Swimlane Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas.
6.		Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.

2.5.3 *Class Diagram*

Class Diagram adalah *Class diagram* mengembangkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem (Rosa dan Salahuddin, 2019). Berikut simbol-simbol yang akan digunakan dalam menggambarkan *Class Diagram* dapat dilihat pada Tabel 2.4.

Tabel 2.4 Simbol *Class Diagram*

No.	Simbol	Deskripsi
1.		Kelas pada struktur sistem**.**

2.	Antar Muka/Interface  Nama_Interface	Sama dengan konsep interface dalam pemrograman berorientasi objek.
3.	Asosiasi / Asociation 	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *symbol*
4.	Asosiasi Berarah / *Digunakan Association* 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *symbol*.
5.	Generalisasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus)
6.	Ketergantungan / dependency 	Relasi antar kelas dengan makna ketergantungan antar kelas.
7.	Agregasi / *aggregation* 	Relasi antar kelas dengan makna semua bagian *(whole-part)*

2.6 Pengujian *Black Box Testing*

Black box testing yaitu pengujian perangkat lunak dari segi pendefinisian fungsional tanpa menguji desain dan kode program. Pengjian dimaksudkan untuk mengetahui apakan fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan (Rosa dan Salahuddin, 2019).

Pengujian yang dilakukan dengan membuat kasus yang bersifat mencoba semua fungsi dengan memakai perangkat lunak apakah sesuai dengan spesifikasi yang dibutuhkan. Kasus uji dilakukan harus dibuat dengan benar dan salah, seperti proses *login* “Jika user memasukan *username* dan *password* yang benar maka dapat *login* ?”.