PENERAPAN *CLEAN CODE* PADA PENGEMBANGAN PERANGKAT LUNAK MANAJEMEN DAN MONITORING LAHAN PERTANIAN PADI

PROPOSAL SKRIPSI

Oleh:

ABU MUSHONNIP

NIM 1805001

PROGRAM STUDI REKAYASA PERANGKAT LUNAK

JURUSAN TEKNIK INFORMATIKA

POLITEKNIK NEGERI INDRAMAYU

FEBRUARI 2022

HALAMAN PENGESAHAN

PENERAPAN *CLEAN CODE* PADA PENGEMBANGAN PERANGKAT LUNAK MANAJEMEN DAN MONITORING LAHAN PERTANIAN PADI

Disusun oleh :

ABU MUSHONNIP

NIM 1805001

Proposal Skripsi disetujui oleh:

Calon Pembimbing : A. Sumarudin, S.Pd., MT., M.Sc.
NIP 198610102019031014

……..……

Indramayu, 28 Juli 2022
Koordinator Program Studi
D4 Rekayasa Perangkat Lunak

Darsih, S.Kom., M.Kom.
NIP 198109062021212004

DAFTAR ISI

Halaman

HALAMAN JUDUL……... i
HALAMAN PENGESAHAN…… ii
DAFTAR ISI……... iii
1. Latar Belakang Masalah…….. 1
2. Rumusan Masalah……... 3
3. Batasan Masalah…….. 3
4. Tujuan…….. 3
5. Manfaat…… 3
6. Tinjauan Pustaka..…… 4
7. Metode Penelitian…… 8
8. Rencana Kegiatan…… 12
DAFTAR PUSTAKA..……. 13

1. Latar Belakang Masalah

Indonesia adalah negara yang dikenal dengan sebutan negara agraris karena sebagian besar penduduknya bermata pencaharian di bidang pertanian. Pada Agustus 2020, Badan Pusat Statistik mencatat penduduk yang bekerja di sektor pertanian sebanyak 38,23 juta orang atau 29,76 persen dari jumlah penduduk bekerja yang jumlahnya 128,45 juta orang (Annur, 2020).

Di Indonesia beras menjadi salah satu makanan pokok yang sangat sulit digantikan, sehingga keberadaan beras menjadi prioritas utama bagi masyarakat dalam memenuhi kebutuhan sehari-hari. Untuk menghasilkan beras yang berkualitas baik, petani harus mengelola tanaman padi sebaik mungkin, mulai dari pemilihan benih, pengelolaan lahan, pemupukan, pengendalian OPT dan lain sebagainya.

Smart farming dapat digunakan para petani karena dapat mempermudah kegiatan pekerjaan para petani salah satunya dalam pemantauan lahan pertanian secara periodik dengan menggunakan sistem akuisisi data. Sistem akuisisi data berfungsi untuk mengambil dan mengumpulkan data dari lingkungan lahan pertanian.

Dengan adanya sistem monitoring tersebut, maka diperlukan adanya sistem manajemen pertanian yang berkelanjutan dan dapat memudahkan petani dalam mengelola lahan beserta perangkat yang tertanam.

Sebuah studi di antara 227 profesional IT menyatakan bahwa keterampilan yang harus dimiliki seorang programmer yaitu kemampuan untuk membaca, memahami, dan memodifikasi kode program yang ditulis oleh orang lain. Kualitas kode yang rendah memberikan pengaruh langsung terhadap perawatan sebuah program (Dietz, 2018).

Clean code merupakan suatu konsep atau petunjuk penulisan struktur kode yang bersih (Clean), dimana kode yang ditulis dapat dibaca oleh pengembang lain dan dapat diubah dengan mudah (Jackson, 1984).

Dengan didapatkannya dugaan masalah tersebut, penulis bermaksud merancang sebuah platform berupa aplikasi web yang memiliki

kode program yang terukur baik sehingga aplikasi tersebut dapat dikembangkan dengan mudah oleh programmer lain. Mengingat pengembangan aplikasi *smart farm* terus dilakukan di lingkungan Politeknik Negeri Indramayu setiap tahunnya, hal ini diharapkan dapat menjadi jawaban untuk pengembangan aplikasi yang berkelanjutan.

2. Rumusan Masalah

Berdasarkan pemaparan latar belakang di atas, maka dapat diidentifikasi masalah pada penelitian ini yaitu:

1. Bagaimana mengimplementasikan clean code pada pengembangan aplikasi web.
2. Seberapa besar pengaruh dari penerapan *clean code* pada kualitas perangkat lunak.

3. Batasan Masalah

Untuk memfokuskan pembahasan, dapat diperoleh beberapa batasan masalah, di antaranya:

1. Implementasi *clean code* dilakukan pada bagian program *backend.*
2. Pengukuran kualitas perangkat lunak dilakukan menggunakan *static analysis tool* yaitu PhpMetrics, dengan melihat nilai *Maintainability Index.*

4. Tujuan

Maksud dilaksanakannya penelitian ini adalah menerapkan konsep *clean code* pada pengembangan aplikasi web. Adapun tujuan dari penelitian ini yaitu mendapatkan rancangan aplikasi yang mempunyai kualitas kode yang baik dan dapat terpelihara.

5. Manfaat

Manfaat yang diperoleh dari penelitian ini adalah memberi gambaran kepada pengembang aplikasi web mengenai bagaimana *clean code* dapat diterapkan.

6. Tinjauan Pustaka

1. Studi Literatur

Pada penelitian pengembangan aplikasi manajemen pertanian, Langkah awal yang dilakukan yaitu menggali sumber informasi yang terkait dengan penelitian-penelitian sebelumnya baik referensi dari buku maupun jurnal.

Proses penggalian literasi dilakukan dengan tujuan untuk mendapatkan informasi yang memiliki hubungan dengan judul terkait. Beberapa studi literatur tersebut sebagai berikut:

1. Penelitian yang dilakukan oleh Daniyal Ahmad Rizaldhi dari Universitas Komputer Indonesia dengan judul "*Implementasi Clean Code Dan Design Pattern Untuk Meningkatkan Maintainability Pada Aplikasi Content Marketing"* pada tahun 2019. Dalam penelitian tersebut penulis menggunakan bahasa pemrograman Javascript (Rizaldhi, 2019).
2. Penelitian yang dilakukan oleh Ilham Prasetyo dari Universitas Komputer Indonesia dengan judul *"Implementasi Clean Code Dan Code Refactoring Untuk Meningkatkan Maintainability Pada Luarsekolah Bot (Lusa Bot)"* pada tahun 2021. Dalam penelitian tersebut, peneiti menggunakan bahasa pemrograman PHP dan *framework* Codeigniter (Prasetyo, 2021).

Dengan adanya tinjauan pustaka di atas, diharapkan pada penelitian kali ini dapat memperbarui penelitian-penelitian sebelumnya.

2. Maintainability Index

Maintainability Index adalah *software metric* yang mengukur sebuah perangkat lunak mudah atau sulit untuk mengalami perawatan atau perubahan di masa mendatang. *Maintainability Index* terdiri atas metrik *Halstead Volume* (HV), metrik *Cyclomatic Complexity* (CC), rata-rata jumlah baris kode per modul (LOC), dan persentase jumlah

komentar per modul (COM) (Stapelberg, 2009) . Adapun formula *Maintainability Index* sebagai berikut:

$$MI = 171 - 5.2 \times \ln(HV) - 0.23 \times CC - 16.2 \times \ln(LOC) + 50 \times \sin\left(\sqrt{2.4 \times COM}\right)$$

Secara umum, nilai dari Maintainability Index diukur dari 0 sampai 100, di mana semakin tinggi nilai tersebut menandakan tingginya maintainability dari kode sumber yang diukur (Chen, Alfayez, Srisopha, Boehm, & Shi, 2017) . Nilai tersebut terbagi menjadi tiga kategori yang dapat dilihat pada Tabel berikut:

Tabel 6.1 Klasifikasi Maintainability Index

Nilai MI	Keterangan
MI > 85	Dapat dipelihara dengan baik
65 < MI ≤ 85	Cukup untuk dapat dipelihara
MI ≤ 65	Sulit untuk dipelihara

3. *Clean Code*

Clean code adalah *code* dalam *software* dengan format yang benar dan disusun dengan rapi dan baik sehingga programmer lain dapat membaca dan memodifikasi *code* tersebut tanpa harus menanyakan terlebih dahulu kepada programmer sebelumnya (Arhandi, Pramitarini, & Alviandra, 2019) . *Clean code* juga bertujuan untuk mengatasi penurunan tingkat produktivitas pengembangan perangkat lunak akibat dari struktur kode yang berantakan.

Adapun hal inti yang dibahas dalam *clean code* menurut (Martin, 2008) diantaranya sebagai berikut:

1. *Meaningful Names*

   Konsep ini memberi petunjuk dalam melakukan penamaan variabel, *method*/fungsi, dan *class*/modul agar lebih mudah dipahami.

a. Nama yang dapat dieja dan memiliki arti

Penggunaan nama harus dapat dieja dan secara jelas menerangkan kegunaan dari kode yang ditulis agar memudahkan pengembang lain memahaminya tanpa memerlukan *effort* yang lebih. Contoh petunjuk ini dapat dilihat pada Gambar 6.1

```
Buruk
const yyyymmdstr = moment().format("YYYY/MM/DD");

Baik
const currentDate = moment().format("YYYY/MM/DD");
```

Gambar 6.1

b. Nama yang mudah dicari

Petunjuk ini bermaksud memudahkan pengembang dalam *tracing* atau mencari kode. Contohnya pada variabel konstan pada Gambar 6.2

```
Buruk
setTimeout(blastOff, 86400000);

Baik
const MILLISECONDS_IN_A_DAY = 86400000;
setTimeout(blastOff, MILLISECONDS_IN_A_DAY);
```

Gambar 6.2

2. *Clean Functions*

Pada konsep ini terdapat petunjuk dalam menulis *method*/fungsi yang bersih agar lebih mudah untuk dipahami. Berikut merupakan petunjuk yang dapat digunakan dalam menulis *method*/fungsi:

a. Jumlah parameter

Dalam menulis method/fungsi yang memiliki parameter, jumlah parameter ideal dari suatu fungsi maksimal sebanyak 2 parameter. Hal tersebut dimaksudkan untuk menghindari banyaknya varian pengujian terhadap method/fungsi yang ditulis.

b. Jumlah pekerjaan

Suatu method/fungsi disarankan untuk melakukan hanya satu pekerjaan saja. Suatu method/fungsi yang melakukan lebih dari satu pekerjaan akan sulit untuk dibuat, diuji, dan dipahami. Method/fungsi yang lebih kecil akan lebih mudah untuk dimodifikasi.

3. *Clean Classess*

   Pada konsep ini terdapat petunjuk dalam membuat *class*/modul yang lebih bersih dan terorganisir. Berikut merupakan petunjuk dalam membuat *class*/modul:

   a. *Single Responsibility Principle*

      Suatu *class*/modul disarankan untuk berukuran tidak terlalu besar, pengembang akan kerepotan dalam memahami *class*/modul tersebut. Contoh petunjuk dapat dilihat pada Gambar 6.3

```
Buruk
class UserSettings {
  constructor(user) {
    this.user = user;
  }
  changeSettings(settings) {
    if (this.verifyCredentials()) {
      // ...
    }
  }
  verifyCredentials() {
    // ...
  }
}
Baik
class UserAuth {
  constructor(user) {
    this.user = user;
  }
  verifyCredentials() {
    // ...
  }
}
class UserSettings {
  constructor(user) {
    this.user = user;
    this.auth = new UserAuth(user);
  }
  changeSettings(settings) {
    if (this.auth.verifyCredentials()) {
      // ...
    }
  }
}
```

Gambar 6.3

4. *Clean Comments*
5. *Clean Error Handling*
6. *Clean Code Formatting*
7. *Clean Object and Data Structures*

7. Metode Penelitian

1. Teknik Pengumpulan Data

Teknik pengumpulan data pada penelitian ini meliputi:

1. Studi Literatur

   Penelitian ini diawali dengan mengkaji referensi-referensi melalui buku, jurnal, laporan-laporan ataupun lainnya yang berkaitan dengan topik penelitian.

2. Observasi

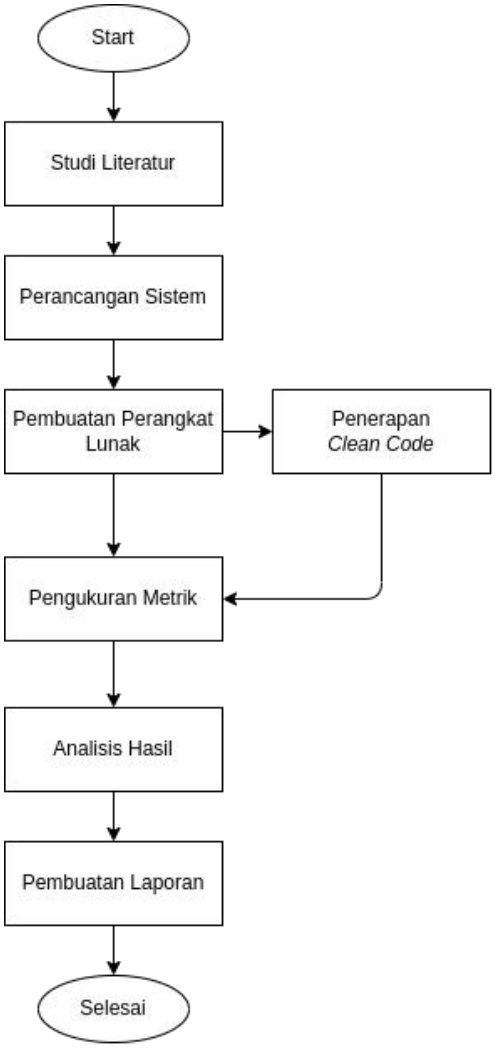
   Observasi merupakan teknik mendapatkan data primer dengan cara mengamati langsung obyek datanya (Jogiyanto, 2008) . Pada penelitian ini, observasi dilakukan pada masyarakat petani untuk mengetahui kebutuhannya.

3. Studi Dokumen

   Peneliti mengandalkan dokumen-dokumen yang terkait dengan pengimplementasian *clean code* pada pengembangan perangkat lunak sebagai salah satu sumber data sebagai penunjang penelitian.

2. Tahapan Penelitian

Adapun tahapan penelitian yang dilakukan adalah sebagai berikut:

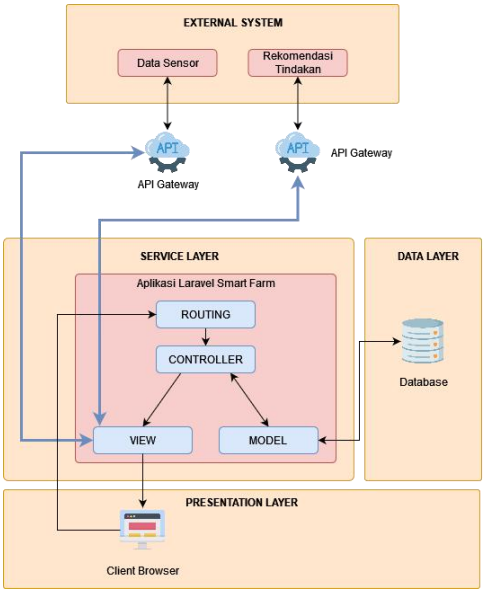


Gambar 7.2

1. Perancangan Sistem

   Pada tahap ini dilakukan penggalian kebutuhan perangkat lunak yang dibutuhkan oleh pengguna maupun arsitektur yang diperlukan.

   a. Perancangan Arsitektur



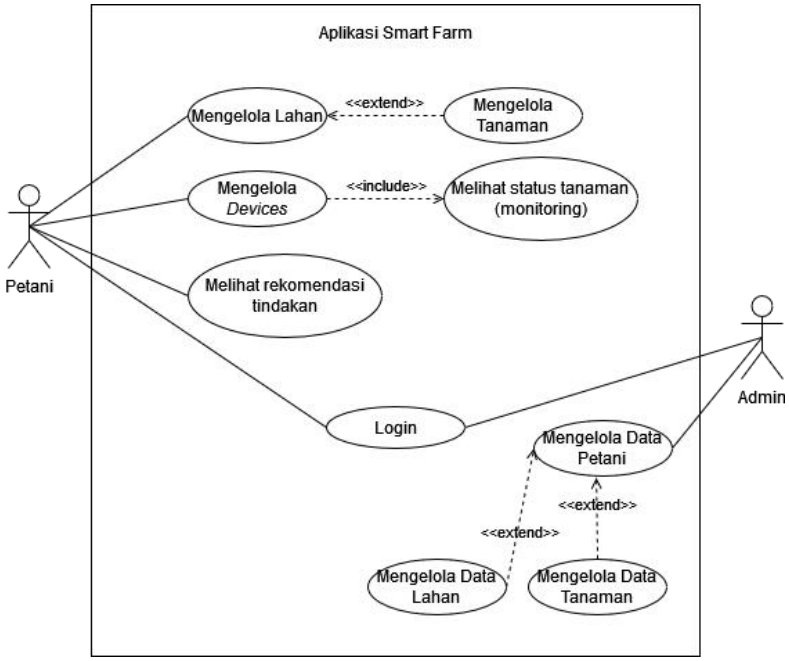
Gambar 7.1 Perancangan Arsitektur Sistem

Gambar 7.1 di atas menjelaskan tentang perancangan arsitektur dari sistem yang akan dibuat. Secara umum sistem yang dibuat memiliki 4 *layer*: (1) *Service Layer* berisikan perangkat lunak utama *frontend* dan *backend* dari Laravel, (2) *Data Layer* berisikan PostgreSQL *database server* yang digunakan untuk menyimpan data pada perangkat lunak utama, (3) *External System* merupakan dependensi perangkat lunak yang berisikan *service* untuk pengambilan data *sensor* dan rekomendasi tindakan, (4) *Presentation Layer* dewasa ini perangkat lunak yang dikembangkan dapat diakses melalui *web browser.*

b. Desain

Pada tahap desain dilakukan perancangan perangkat lunak menggunakan pemodelan UML *(Use Case Diagram, Activity Diagram, dan Class Diagram*).

1. *Use Case Diagram*



Gambar 7.2 *Use Case Diagram* Aplikasi Smart Farm

Gambar 7.3 di atas merupakan rancangan awal Use Case Diagram dari aplikasi Smart Farm.

2. Penulisan Kode Program dengan Teknik *Clean Code*

   Pada tahap ini dilakukan penulisan kode program tanpa dengan mengimplementasikan teknik *clean code* dan dengan mengimplementasikan *clean code* menggunakan bahasa pemrograman PHP dengan *framework* Laravel.

3. Analisa Hasil Metrik

   Hasil kode program yang dibuat akan dihitung nilai *Maintainability Index*nya, kemudian dilakukan analisis perhitungan sebelum dan sesudah penerapan *clean code*.

3. Metode Clean Code

Adapun penerapan konsep *clean code* maupun *design pattern* yang dilakukan merupakan eksperimen, sehingga tidak ada konsep spesifik yang diusulkan.

8. Rencana Kegiatan

Tabel 8.1 Rencana Kegiatan

No	Kegiatan	Februari				Maret				April				Mei				Juni				Juli				Agustus			
		I	II	III	IV	I	II	III	IV	I	II	III	IV	I	II	III	IV	I	II	III	IV	I	II	III	IV	I	II	III	IV
1	Pembuatan Proposal																												
2	Identifikasi Masalah																												
3	Analisis Kebutuhan Sistem																												
4	Studi Literatur																												
5	Membuat Rancangan Sistem																												
6	Implementasi Program																												
7	Pengujian Sistem (Testing)																												
8	Revisi Konsep, Desain Rancangan, Kode Program																												
9	Penyusunan Laporan Penulisan Skripsi																												
10	Pelaksanaan Sidang Skripsi																												
11	Pelaksanaan Revisi Skripsi																												

DAFTAR PUSTAKA

Annur, C. M. (2020). *Sektor Pertanian Paling Banyak Menyerap Tenaga Kerja Indonesia*. Diambil kembali dari Katadata: https://databoks.katadata.co.id/datapublish/2020/11/12/sektor-pertanian-paling-banyak-menyerap-tenaga-kerja-indonesia

Arhandi, P. P., Pramitarini, Y., & Alviandra, R. (2019). Desain Prototype Frontend Auto Generator Based On REST API. *Seminar Informatika Aplikatif Polinema*, 389-393.

Chen, C., Alfayez, R., Srisopha, K., Boehm, B., & Shi, L. (2017). Why Is It Important to Measure Maintainability and What Are the Best Ways to Do It? *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*.

Dietz, L. W. (2018). Teaching Clean Code. *Proceedings of the 1st Workshop on Innovative Software Engineering Education*, 24-27.

Jackson, M. (1984). ISO/IEC.. Systems and software engineering-Systems and software Quality Requirements and Evaluation (SQuaRE)-System and softwre quality models. *Switz. ISOIEC Towar. a Syst. Syst. Methodol. J. Oper. Res. Soc*, 473-486.

Jogiyanto. (2008). *Metodologi Penelitian Sistem Informasi.* Yogyakarta: Andi.

Martin, R. C. (2008). *Clean Code: A Handbook of Agile Software Craftsmanship.* Boston: Pearson.

Prasetyo, I. (2021). *Implementasi Clean Code Dan Code Refactoring Untuk Meningkatkan Maintainability Pada Luarsekolah Bot (Lusa Bot).*

Rizaldhi, D. A. (2019). *Implementasi Clean Code Dan Design Pattern Untuk Meningkatkan Maintainability Pada Aplikasi Content Marketing.*

Stapelberg, R. F. (2009). *Handbook of Reliability, Availability, Maintainability and Safety in Engineering Design.* London: Springer.