Jurnal Sains Komputer & Informatika (J-SAKTI)
Volume 7 Nomor 2, September 2023, pp. 686-697
ISSN: 2548-9771/EISSN: 2549-7200
https://tunasbangsa.ac.id/ejurnal/index.php/jsakti

Penggunaan *Factory Method Design Pattern* Pada *Framework* Flask di Dalam Aplikasi *Dashboard*

Devin William Sumbaluwu[1], Ramos Somya[2]
[1,2]Teknik Informatika, Universitas Kristen Satya Wacana, Indonesia
e-mail: [1]672018282@student.uksw.edu, [2]ramos.somya@uksw.edu

Abstract

Problems related to the use of an object of Flask for various environments are often caused by application designs that are less effective and efficient. An object of Flask is used to run a Flask application based on the states or application environments. Each state or environment has different configuration requirements. The factory method design pattern solves the design problem by separating functions that return an object of Flask based on another object of the configuration class. The implementation of the factory method design pattern will be realized in a dashboard website. It will be used on the backend so that there will be a unit test code to prove the app can be run in a different state or environment.

Keywords*: Factory Method, Flask, Python, TDD, Dashboard*

Abstrak

Permasalahan terkait penggunaan objek dari kelas Flask terhadap berbagai macam kondisi sering kali disebabkan oleh desain aplikasi yang kurang efektif dan efisien. Objek dari kelas Flask digunakan untuk menjalan aplikasi Flask berdasarkan kondisi atau environtment aplikasi. Setiap kondisi atau environtment mempunyai kebutuhan konfigurasi yang berbeda - beda. Penggunaan factory method design pattern dapat menyelesaikan masalah desain dengan cara memisahkan fungsi yang mengembalikan nilai objek dari kelas Flask berdasarkan objek dari kelas konfigurasi tertentu. Implementasi dari penggunaan factory method design pattern akan diwujudkan di dalam aplikasi dashboard berbasis website. Penerapan factory method design pattern akan digunakan pada sisi backend, sehingga akan ada penulisan kode unit test untuk membuktikan bahwa kode dapat dijalankan di environtment yang berbeda yaitu testing.

Kata kunci*: Factory Method, Flask, Python, TDD, Dashboard*

1. PENDAHULUAN

Pada proses pembuatan aplikasi pengembang sering kali menemukan desain aplikasi yang kurang efektif, sehingga setiap kali terjadi perubahan di komponen-komponen tertentu maka akan berdampak pada fitur-fitur yang lainnya. Hal ini memaksa pengembang untuk melakukan perubahan atau bahkan merancang kembali aplikasi tersebut. Cara tersebut sangatlah tidak efektif dan efisien dari segi waktu dan biaya. Perubahan sekecil apapun sebaiknya tidak memberikan dampak yang signifikan terhadap bagian-bagian yang lain pada sebuah aplikasi. Salah satu cara yang cukup sering digunakan untuk menyelesaikan permasalahan tersebut yaitu mengimplementasi design pattern. Design pattern merupakan sebuah solusi desain terhadap permasalahan *object-oriented design* yang terjadi berkali-kali dengan cara merepresentasikan inti atau konteks dari permasalahan tersebut di dalam diagram - diagram UML yang selanjutnya dapat diimplementasikan dengan cara yang berbeda-beda berdasarkan permasalahan ataupun bahasa pemrograman yang digunakan [1]. Design pattern memanfaatkan elemen-elemen dalam *object-*

oriented design seperti kelas dan objek beserta peran dan kolaborasi antar elemen agar menghasilkan solusi *object-oriented design* yang dapat digunakan kembali [2]. Berdasarkan hal itu, design pattern dapat diaplikasikan di dalam bahasa pemrograman yang berorientasi objek seperti python.

Manfaat dari *design pattern* dapat diterapkan pada pengembangan *website* menggunakan *framework* python yaitu Flask. Permasalahan terkait pemanggilan dan penggunaan objek dari kelas Flask untuk berbagai macam kondisi atau *environment* dapat diselesaikan menggunakan *factory method design pattern*. *Factory method design pattern* merupakan salah satu *creational design pattern* yang menggunakan *factory method* untuk menyelesaikan permasalahan terkait pembuatan objek pada method tanpa menentukan secara spesifik tipe kelas dari objek yang akan dibuat. *Factory method* memberikan keleluasaan kepada kelas turunannya atau kelas yang mengimplementasinya untuk menentukan tipe kelas dari *factory method* tersebut.

Pada *framework* Flask, pembuatan *instance* objek dari kelas Flask dibuat di dalam method. Cara ini dikenal sebagai *application factory*. *Application factory* memiliki kesamaan dengan *factory method design pattern* yaitu pembuatan objek dilakukan di dalam *method*. Namun, cara ini menimbulkan masalah saat ingin membuat objek *Flask* yang membutuhkan *file* konfigurasi tertentu berdasarkan kondisi yang sedang dijalankan. Setiap kondisi memiliki kebutuhan konfigurasi yang berbeda seperti akses URL database untuk tahap *testing* berbeda dengan tahap *production* begitupun dengan tahap *development*. Dalam kasus ini, *Factory Method* digunakan untuk memisahkan *method* yang mengembalikan nilai objek dari kelas Flask berdasarkan *instance* dari objek kelas konfigurasi tertentu.

Beberapa penelitian terdahulu yang dijadikan acuan untuk melakukan penelitian ini membahas tentang implementasi dari *Factory Method Design Pattern*, penggunaan *framework* Flask untuk pembuatan aplikasi tertentu, dan penggunaan *test-driven development* pada proses pengembangan aplikasi. Banyak dari penelitian tersebut fokus pada penerapan atau implementasi terhadap teknologi terkait. Membuat atau mengusulkan perbaikan terhadap metode atau teknologi tertentu pada pengembangan aplikasi merupakan hal yang menarik untuk dijadikan topik penelitian. Perbaikan metode atau teknologi pengembangan aplikasi juga dapat diimplementasikan ke dalam bentuk aplikasi tertentu agar dapat membuktikan keberhasilan dari perbaikan metode tersebut. Berdasarkan alasan itu, penelitian ini membutuhkan berbagai macam penelitian terdahulu baik yang membahas mengenai metode pengembangan aplikasi, penggunaan teknik pengembangan tertentu, ataupun implementasi dari teknologi atau framework tertentu.

Penelitian yang berjudul "Human Resource Management System Based on Factory Method Design Pattern" membahas tentang pembuatan sistem atau aplikasi manajemen sumber daya manusia di dalam perusahaan dengan memanfaatkan *factory method design pattern*. Permasalahan yang tuliskan dalam penelitian itu terdapat sedikit perbedaan di setiap antarmuka modul sehingga memungkinkan terjadinya penggunaan kode yang sama di dalam banyak antarmuka. Penggunaan kode yang sama di dalam banyak antarmuka menyulitkan

pengembang karena pengembang harus menuliskan beberapa kode secara berulang, hal ini sangat menyita waktu. Di dalam penelitian tersebut dilampirkan sebuah gambar diagram yang menunjukan tahapan penulisan kode antarmuka merupakan tahapan yang paling banyak menyita waktu. Hasil dari penelitian itu, pemanfaatan factory method design pattern dapat meningkatkan *reusability* pada sistem, sehingga desainer tidak perlu fokus terhadap perubahan-perubahan yang terjadi di berbagai macam tampilan, desainer hanya perlu fokus pada atribut-atribut tertentu [3].

Penelitian selanjutnya yang berjudul "An approach for applying Test-Driven Development (TDD) in the development of randomized algorithms" yang menerapkan metode *test-driven development* di dalam pengembangan algoritma *randomized*. Hasil dari penelitian ini mengajukan sebuah pendekatan yang dapat memanfaatkan TDD dalam tahap pengembangannya dan merupakan teknik desain aplikasi dengan melibatkan algoritma *randomized*. Pendekatan tersebut terdiri dari perluasan TDD yang implementasinya atau aplikasinya dapat digunakan untuk pengembangan algoritma yang memiliki karakteristik *non-deterministic* salah satunya algoritma *randomized* [4].

Pada penelitian yang berjudul "Design an MVC Model using Python for Flask Framework Development" dibahas mengenai penggunaan paradigma MVC di dalam *framework* Flask. Di dalam penelitian itu dibahas mengenai *framework* Flask yang pada dasarnya tidak menerapkan paradigma atau pendekatan MVC di dalam penerapannya. Sehingga, tujuan dari penelitian itu yaitu membuat sistem yang dapat menerapkan model MVC di dalam *framework* Flask. Dalam penerapannya dibantu oleh *generator* yang dapat membuat *folder* dan *file* terstruktur secara otomatis. Hasil dari penelitian itu menyebutkan bahwa sistem yang telah dibuat dapat membantu pengembang untuk meningkatkan kecepatan dan kualitas pekerjaannya. Sistem juga menyediakan sebuah framework dan platform kerja yang mendukung baik untuk pemula maupun yang berpengalaman [5].

Dari ketiga penelitian tersebut dapat menjadi acuan untuk melakukan penelitian mengenai penerapan *factory method design pattern* di dalam *framework* Flask. Adapun penelitian-penelitian sebelumnya yang tidak disebutkan seperti di atas juga belum pernah ada penelitian mengenai penggunaan *design pattern* untuk meningkatkan kinerja framework tertentu. Oleh karena itu, tiga penelitian yang telah disebutkan dapat menjadi referensi untuk melakukan penelitian tentang penerapan *factory method design pattern* di dalam *framework* Flask dengan bantuan teknik *test-driven development* yang diimplementasikan di dalam aplikasi *dashboard* berbasis website.

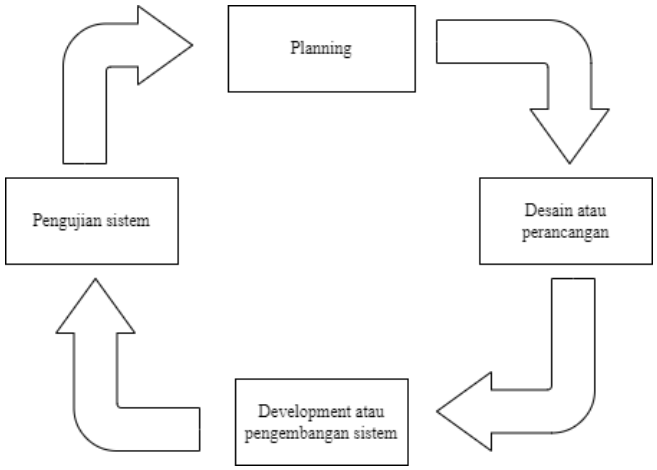
Aplikasi *dashboard* yang dibuat akan menyajikan informasi – informasi mengenai data akademik dan keuangan mahasiswa. Aplikasi *dashboard* bertujuan untuk membantu beberapa pejabat universitas dalam mengambil sebuah keputusan berdasarkan informasi yang tersedia di dalam *dashboard*. Oleh karena itu, *framework* Flask yang telah menerapkan *factory method* akan dibuktikan perannya di dalam aplikasi *dashboard*.

Jurnal Sains Komputer & Informatika (J-SAKTI)
Volume 7 Nomor 2, September 2023, pp. 686-697
ISSN: 2548-9771/EISSN: 2549-7200
https://tunasbangsa.ac.id/ejurnal/index.php/jsakti

2. METODOLOGI PENELITIAN

2.1. Metode Pengembangan Sistem

Metode yang digunakan dalam pengembangan website dashboard adalah *agile development*. Metode ini menekankan pada proses pengembangan aplikasi yang berulang dan adaptif terhadap perubahan – perubahan pada sistem atau aplikasi yang dibangun [6]. Salah satu keuntungan menggunakan metode agile yaitu dapat menerapkan continuous testing. Continuous testing merupakan metode pengujian perangkat lunak yang dilakukan secara otomatis dan terus-menurus selama proses pengembangan [7], [8]. Dalam hal ini, teknik pengembangan TDD (Test-Driven Development) dapat diterapkan dengan menggunakan metode pengembangan ini.



Gambar 1. Siklus *Agile Development*

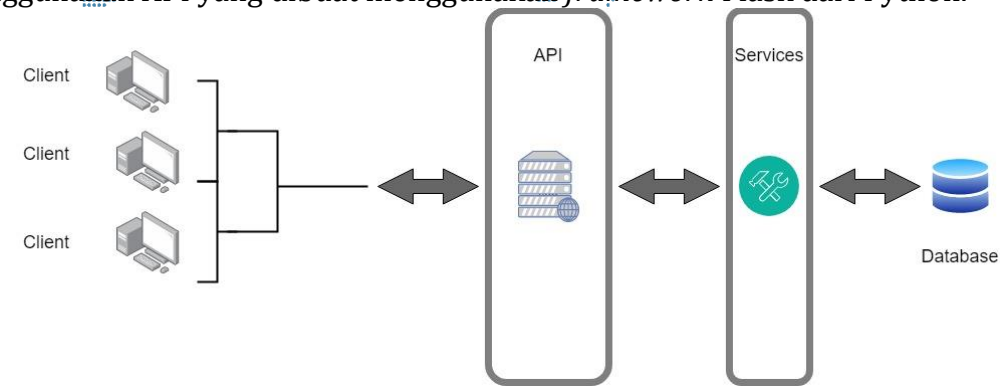
Berikut merupakan penjelasan mengenai tahapan di dalam siklus *Agile Development* tahapan di dalam yang pada Gambar 1.

1) Pada tahap *planning* akan dilakukan penentuan *requirement* mengenai pembuatan aplikasi, seperti hal-hal apa saja yang harus ada dalam proses pengembangan aplikasi.
2) Tahap desain atau perancangan merupakan proses perancangan *database*, pembuatan UML, dan pembuatan tampilan atau user interface (UI) aplikasi.
3) Dalam tahap development atau pengembangan sistem akan dilakukan pengembangan aplikasi diantaranya *coding* dan penulisan *script* kode *unit test*. Kode *unit test* dituliskan pada tahap ini, karena dalam pengembangan aplikasi menerapkan pendekatan *test-driven development* [9].
4) Tahap *testing* yang dilakukan pada tahap ini berbeda dengan yang dilakukan pada tahap *development* dimana pada tahap ini melibatkan pihak eksternal yaitu dari sisi perspektif user. User memeriksa efek yang terlihat secara eksternal, seperti menentukan *output* yang benar dari suatu sistem dengan *input* tertentu [10]. Tahap ini juga dikenal dengan *acceptance test*.

Jurnal Sains Komputer & Informatika (J-SAKTI)
Volume 7 Nomor 2, September 2023, pp. 686-697
ISSN: 2548-9771/EISSN: 2549-7200
https://tunasbangsa.ac.id/ejurnal/index.php/jsakti

2.2. Arsitektur Sistem

Sistem aplikasi dibagi menjadi 2(dua) sisi yaitu *client*(*frontend*) dan *server*(*backend*). Sisi *client* dibangun menggunakan React JS, sedangkan sisi *server* menggunakan API yang dibuat menggunakan *framework* Flask dari Python.



Gambar 2. Arsitektur Sistem

Pada Gambar 2 dijelaskan bahwa *client* dapat menampilkan informasi secara visual dalam bentuk *dashboard* dengan memanfaatkan data yang tersimpan di dalam *database*. Proses komunikasi antara *client* dan *database* melalui pihak perantara yaitu API. Proses komunikasi yang terjadi menggunakan protokol HTTP, API akan menerima *request* dari client dan akan mengidentifikasi tindakan yang harus diambil berdasarkan HTTP *method* yang ada di dalamnya. Sehingga, *request* dapat diteruskan kepada *server* untuk menjalankan *service* tertentu. Adapun beberapa service yang ditunjukan seperti pada Gambar 2 dapat dijalankan berdasarkan format url yang tepat beserta HTTP *method* yang diberikan. Hasil dari proses transaksi antara *client* dan *database* berupa data dengan format json. Selanjutnya, *client* atau sisi *frontend* dapat memanfaatkan data tersebut untuk kebutuhan tampilan.

2.3. Perancangan Sistem

Desain dan perancangan *factory method design pattern* pada *framework* Flask dimulai dari memisahkan kelas dari *config file* yang berisi pengaturan-pengaturan berdasarkan *environtment* aplikasi. Misalnya, pada kasus *development* diharuskan untuk menginisiasi variabel "DEBUG" dengan nilai boolean yaitu True. Sebaliknya, pada kasus *production* variabel "DEBUG" akan diinisiasi menjadi False. Hal ini menyebabkan program akan memastikan bahwa sebuah *file* dari kelas konfigurasi akan mengaktifkan mode debug berdasarkan *environtment* tertentu.

Pada sistem ini, *file* config terdiri dari 4 kelas yaitu sebuah kelas config sebagai kelas parent dan kelas ProductionConfig, DevelopmentConfig, dan TestingConfig sebagai kelas child. Berikut merupakan gambar mengenai kelas – kelas konfigurasi di python.

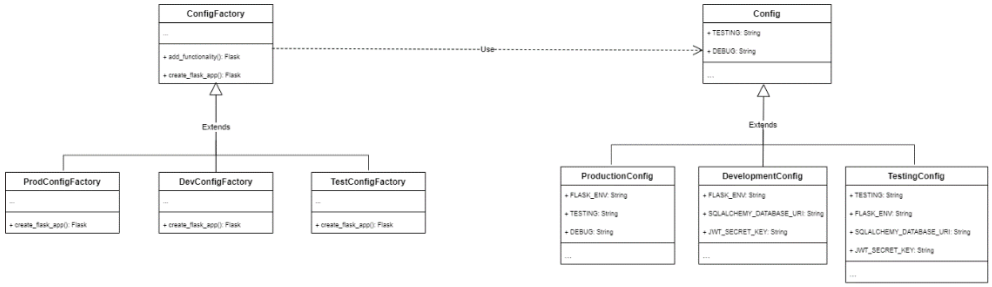
Jurnal Sains Komputer & Informatika (J-SAKTI)
Volume 7 Nomor 2, September 2023, pp. 686-697
ISSN: 2548-9771/EISSN: 2549-7200
https://tunasbangsa.ac.id/ejurnal/index.php/jsakti

config.py > ...

```python
import secrets

class Config:
    TESTING = False
    DEBUG = True
class ProductionConfig(Config):
    FLASK_ENV = 'production'
    TESTING = False
    DEBUG = False

class DevelopmentConfig(Config):
    FLASK_ENV = 'development'
    SQLALCHEMY_DATABASE_URI =  "postgresql://postgres:devin_1234@localhost/db_thesis_test"
    JWT_SECRET_KEY = secrets.token_hex(16)

class TestingConfig(Config):
    TESTING = True
    FLASK_ENV = 'testing'
    SQLALCHEMY_DATABASE_URI =  "postgresql://postgres:devin_1234@localhost/db_unit_test"
    JWT_SECRET_KEY = secrets.token_hex(16)
```

Gambar 3. Kelas-kelas yang berhubungan dengan konfigurasi

Desain *factory method design pattern* yang diterapkan akan digambarkan pada kelas diagram. Kelas diagram menggambarkan pemaanfaatan objek ataupun atribut 1(satu) kelas terhadap kelas yang lainnya. Berikut merupakan kelas *diagram* dari *factory method design pattern* pada *framework* Flask.

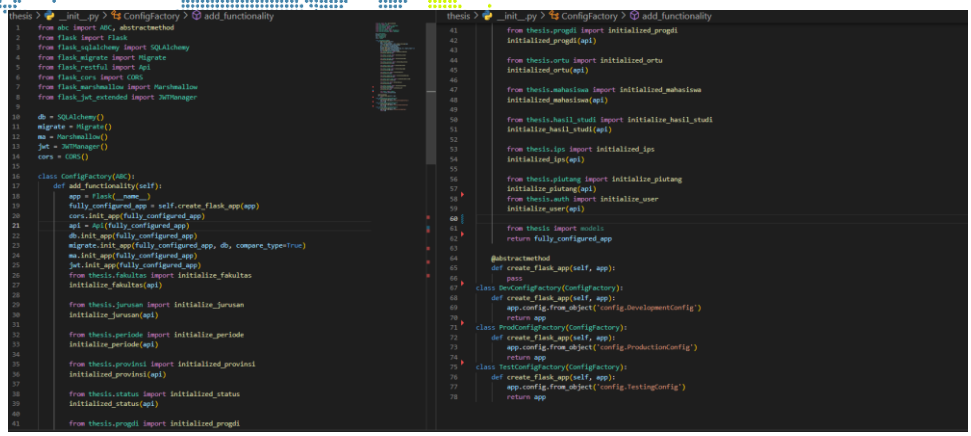


Gambar 4. Kelas *Diagram factory method* pada sistem

Berdasarkan Gambar 4, *method* create_flask_app() yang akan berfungsi sebagai *factory method*. *Method* tersebut berada pada kelas *abstract* yang selajutnya akan diturunkan pada kelas *child* untuk menentukan isi dari *factory method* tersebut. *Interface* ataupun kelas *abstract* sering kali mewarisi *factory method* pada kelas *child* agar dapat mengembalikan nilai objek dari kelas *concrete*. Sehingga, *factory method* pada kelas *abstract* hanya bisa menentukan tipe data dari nilai kembalian *method* menggunakan sebuah *interface* yang selanjutnya akan diimplementasikan oleh kelas lain.

Pada penelitian ini, penggunaan *factory method* akan sedikit berbeda karena kelas *child* yang diwarisi *factory method* oleh kelas *abstract* tidak akan mengembalikan nilai objek kelas *concrete* dari sebuah *interface*, melainkan mengembalikan objek kelas *concrete* yang tidak mengimplementasi kelas manapun. Berikut merupakan potongan *script* kode python mengenai penerapan *factory method*.

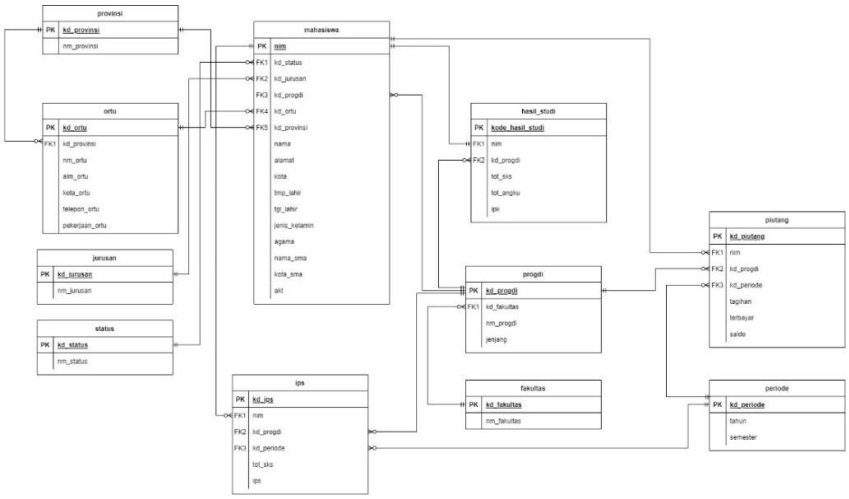
Jurnal Sains Komputer & Informatika (J-SAKTI)
Volume 7 Nomor 2, September 2023, pp. 686-697
ISSN: 2548-9771/EISSN: 2549-7200
https://tunasbangsa.ac.id/ejurnal/index.php/jsakti



Gambar 5. Penerapan *factory method*

Berdasarkan potongan *script* kode, kelas *child* akan mengimplementasi *factory method* dengan menambahkan atau memasukkan objek config ke dalam *argument* pada *method* dari objek kelas Flask. Oleh karena itu, *factory method* akan tetap mengembalikkan objek kelas Flask yang sama pada saat proses *instantiation* objek Flask di dalam method add_functionality().

Factory method design pattern pada *framework* Flask diimplementasikan di dalam *dashboard*. Perancangan aplikasi *dashboard* membutuhkan rancangan ERD basis data yang baik, sehingga dapat menentukan fitur – fitur yang dibutuhkan di dalam aplikasi *dashboard*. Perancangan sistem pada tahap implementasi akan dibahas dalam bentuk ERD berikut.



Gambar 6. ERD Sistem

Desain ERD menggunakan pendekatan Galaxy Schema. Pada *schema* ini tabel dibagi menjadi dua jenis yaitu tabel fakta dan tabel dimensi. Tabel – tabel yang dikategorikan tabel fakta meliputi tabel hasilstudi, tabel mahasiswa, tabel ips, tabel piutang, sedangkan tabel – tabel yang dikategorikan tabel dimensi antara lain tabel progdi, tabel fakultas, tabel ortu, tabel status, tabel periode, tabel provinsi, dan tabel jurusan.

Jurnal Sains Komputer & Informatika (J-SAKTI)
Volume 7 Nomor 2, September 2023, pp. 686-697
ISSN: 2548-9771/EISSN: 2549-7200
https://tunasbangsa.ac.id/ejurnal/index.php/jsakti

3. HASIL DAN PEMBAHASAN

3.1. Pengantar

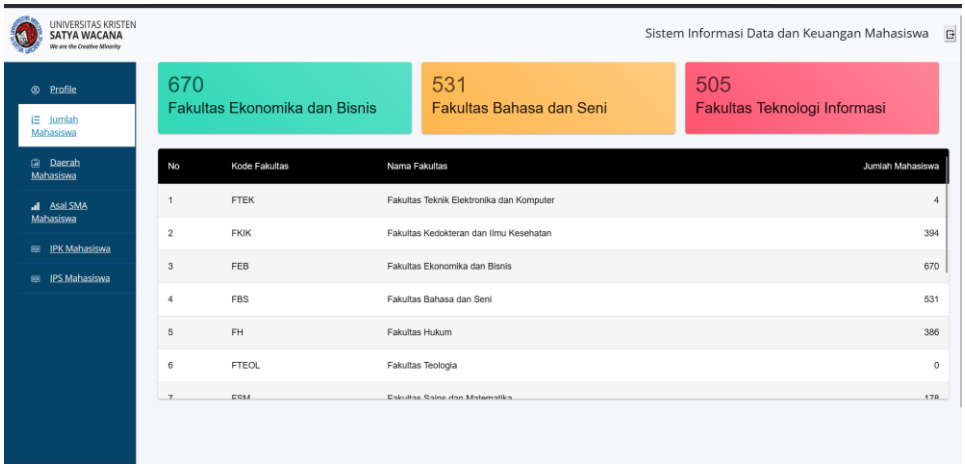
Pada bagian ini akan dibahas mengenai hasil dari pemanfaatan *factory pattern design pattern* pada *framework* Flask di dalam aplikasi *dashboard*. Bagian selanjutnya akan membahas mengenai aplikasi *dashboard* yang memuat fitur – fitur berbeda sesuai dengan hak akses yang telah diterapkan di dalamnya.

3.2. Implementasi Sistem

Hasil implementasi berupa sebuah sistem informasi dalam bentuk aplikasi *dashboard* yang akan menampilkan informasi – informasi akademik universitas. Terdapat 4 (empat) hak akses di dalam aplikasi ini, antara lain Pembantu Rektor 1 (PR 1), Pembantu Rektor 2 (PR 2), Dekan Fakultas, dan Administrator.

1) Pembantu Rektor 1 (PR 1)

Menu – menu yang dapat diakses oleh PR1, antara lain jumlah mahasiswa, daerah mahasiswa, asal SMA mahasiswa, IPK mahasiswa, dan IPS mahasiswa. Berikut merupakan salah satu menu dari *user* PR1 yaitu jumlah mahasiswa.



Gambar 7. Tampilan menu Jumlah Mahasiswa

2) Pembantu Rektor 2 (PR 2)

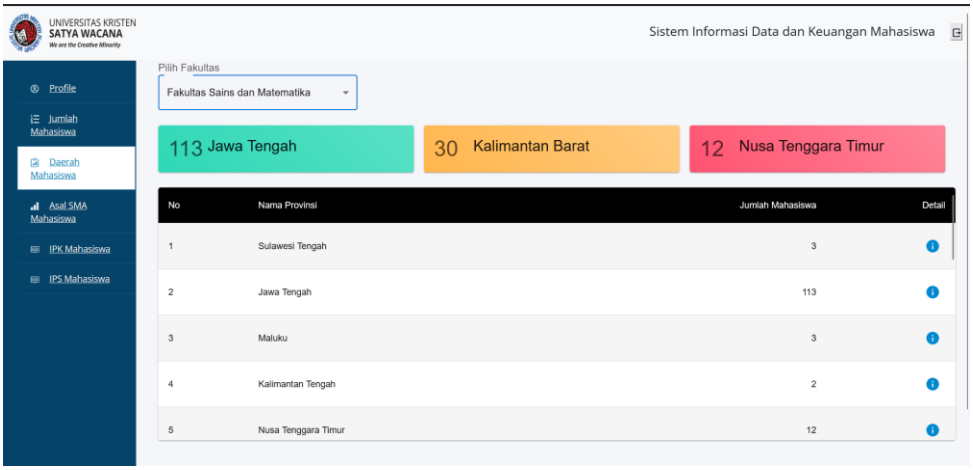
Pembantu rektor 2 (PR 2) dapat mengakses 2 (dua) menu yang berkaitan dengan piutang mahasiswa atau yang berkaitan dengan informasi keuangan mahasiswa. Menu – menu yang dapat diakses oleh PR 2, antara lain informasi piutang tagihan mahasiswa dimensi fakultas dan informasi piutang tagihan – terbayar mahasiswa dimensi fakultas. Salah satu menu yang dapat di akses PR 2 yaitu informasi piutang tagihan – terbayar mahasiswa dimensi fakultas ditampilkan pada Gambar 8.

Jurnal Sains Komputer & Informatika (J-SAKTI)
Volume 7 Nomor 2, September 2023, pp. 686-697
ISSN: 2548-9771/EISSN: 2549-7200
https://tunasbangsa.ac.id/ejurnal/index.php/jsakti

Gambar 8. Tampilan menu piutang tagihan-terbayar

3) Dekan Fakultas

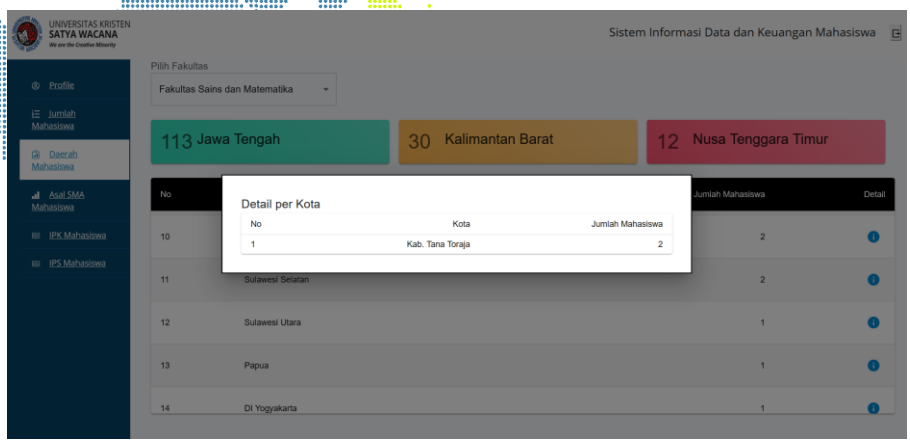
Dekan fakultas dapat mengakses informasi mengenai data akademik mahasiswa dengan dimensi program studi menggunakan filter fakultas. Dekan fakultas menggunakan menu *drop-down* yang berisi daftar fakultas yang ada pada universitas. Menu – menu yang dapat diakses oleh dekan fakultas sama seperti menu – menu yang dapat diakses oleh PR 1, akan tetapi perbedaanya ada pada filter fakultas. Berikut merupakan contoh tampilan menu informasi mengenai daerah asal mahasiswa untuk setiap provinsi sesuai dengan fakultasnya.



Gambar 9. Tampilan menu daerah mahasiswa

Dekan fakultas juga dapat mengakses kota dari setiap provinsi yang tersedia menggunakan fitur detail yang telah disediakan seperti gambar di bawah ini.

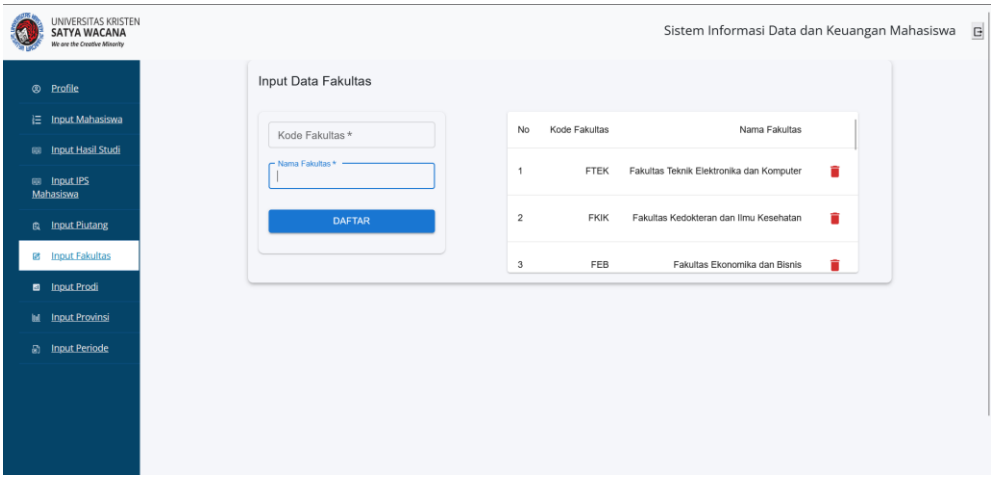
Jurnal Sains Komputer & Informatika (J-SAKTI)
Volume 7 Nomor 2, September 2023, pp. 686-697
ISSN: 2548-9771/EISSN: 2549-7200
https://tunasbangsa.ac.id/ejurnal/index.php/jsakti



Gambar 10. Tampilan menu detail daerah mahasiswa

4) Administrator

Pengguna dengan hak akses sebagai administrator dapat mengakses 8 (delapan) menu yaitu menu input mahasiswa, input hasil studi, input IPS, input piutang, input fakultas, input prodi, input provinsi, dan input periode. Salah satu menu yang dapat diakses oleh Administrator ditampilkan seperti gambar dibawah ini.



Gambar 11. Menu input data fakultas

3.3. Pengujian Sistem

Pengujian yang dilakukan hanya berfokus pada sisi *backend* sebagaimana yang sudah dituliskan mengenai tujuan penelitian yaitu menghasilkan solusi desain yang mampu memisahkan penggunaan objek dari kelas Flask sesuai dengan kebutuhan. Dalam hal ini, kebutuhan akan objek dari kelas Flask yang dapat menjalankan beberapa *dependencies* untuk kebutuhan testing.

Pada proses ini, pengujian dilakukan pada level unit atau komponen. Pengujian menggunakan *framework* python yaitu pytest. Kode yang dituliskan menguji fungsi yang berkaitan dengan proses komunikasi antara *client* dan *backend server*. Hasil dari pengujian akan disajikan di dalam Tabel 1.

Jurnal Sains Komputer & Informatika (J-SAKTI)
Volume 7 Nomor 2, September 2023, pp. 686-697
ISSN: 2548-9771/EISSN: 2549-7200
https://tunasbangsa.ac.id/ejurnal/index.php/jsakti

Tabel 1. Tabel hasil *unit test*

Nama fungsi	**Aksi**	**Status**
test_config_development	Menguji *environtment* atau fase yang sedang dijalankan yaitu *development*	Passed
test_config_testing	Menguji *environtment* atau fase yang sedang dijalankan yaitu *testing*.	Passed
test_login_user	Menguji user apakah berhasil melakukan *login* atau tidak	Passed
test_err_login_with_wrong_email	Menguji user salah memasukkan *email* saat *login*	Passed
test_err_login_with_wrong_password	Menguji user salah memasukkan *password* saat *login*	Passed
test_err_register_user_with_existing_username_or_email	Menguji user saat memasukkan *username* atau *email* yang sudah didaftarkan	Passed
test_register_user	Menguji user saat berhasil melakukan registrasi	Passed
test_get_all_fakultas	Menguji permintaan mengenai data semua fakultas	Passed
test_get_index_fakultas	Menguji permintaan mengenai 1(satu) data fakultas berdasarkan kode fakultas	Passed
test_post_fakultas	Menguji keberberhasilan saat menambahkan data fakultas	Passed
test_put_fakultas	Menguji keberberhasilan saat memperbarui data fakultas	Passed
test_delete_fakultas	Menguji keberberhasilan saat menghapus data fakultas	Passed
test_post_hasil_studi	Menguji keberberhasilan saat menambahkan data hasil studi	Passed
test_post_ips	Menguji keberberhasilan saat menambahkan data IPS	Passed
test_post_mahasiswa	Menguji keberberhasilan saat menambahkan data mahasiswa	Passed
test_get_all_periode	Menguji permintaan mengenai data semua periode	Passed
test_post_periode	Menguji keberberhasilan saat menambahkan data periode	Passed
test_delete_periode	Menguji keberberhasilan saat menghapus data periode	Passed
test_post_piutang	Menguji keberberhasilan saat menambahkan data piutang	Passed
test_post_progdi	Menguji keberberhasilan saat menambahkan data program studi	Passed
test_get_all_progdi	Menguji permintaan mengenai data semua program studi	Passed
test_put_progdi	Menguji keberberhasilan saat memperbarui data program studi	Passed
test_delete_progdi	Menguji keberberhasilan saat menghapus data program studi	Passed
test_get_all_provinsi	Menguji permintaan mengenai data semua provinsi	Passed
test_post_provinsi	Menguji keberberhasilan saat menambahkan data provinsi	Passed
test_put_provinsi	Menguji keberberhasilan saat memperbarui data provinsi	Passed
test_delete_provinsi	Menguji keberberhasilan saat menghapus data provinsi	Passed

4. SIMPULAN

Berdasarkan penelitian yang telah dilakukan, maka dapat disimpulkan bahwa solusi yang diajukan berhasil menghasilkan solusi desain yang dapat memisahkan pembuatan objek dari kelas Flask dengan bantuan *factory method design pattern* berdasarkan kondisi yang terjadi. Proses inisialisasi dilakukan di dalam kelas *parent* yang selanjutnya akan diwariskan kepada kelas *child* untuk menambahkan objek dari kelas konfigurasi ke dalam objek dari kelas Flask. Ada

dua jenis kondisi yang telah dilakukan yaitu kondisi saat tahapan pengembangan (development) dan pengujian (testing). Objek dari kelas Flask akan digunakan di berbagai macam kondisi tergantung kondisi yang sedang dijalankan. Oleh karena itu, penggunaan *factory method design pattern* pada framework Flask terbukti dapat menghasilkan solusi desain yang dapat diterapkan di dalam aplikasi dashboard.

DAFTAR PUSTAKA

[1] B. Bach *et al.*, "Dashboard Design Patterns", *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 1, pp. 342–352, 2023.

[2] M. Aniche, J. W. Yoder, and F. Kon, "Current Challenges in Practical Object-Oriented Software Design", *IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, pp. 113–116, 2019.

[3] X. Xu, H. Hu, N. Hu, L. Xiao, and W. Ying, "Human Resource Management System Based on Factory Method Design Pattern", *International Conference on Information Computing and Applications*, pp. 437–444, 2012.

[4] I. André A. S., G. Eduardo M., P. Sandy M., C. Joelma, and Q. Marcos G., "An approach for applying Test-Driven Development (TDD) in the development of randomized algorithms", *Journal of Software Engineering Research and Development*, vol. 9, no. 6, pp. 1–31, 2018.

[5] M. R. Mufid, M. U. H. Al Rasyid, I. F. Rochimansyah, and A. Rokhim, "Design an MVC Model using Python for Flask Framework Development", *International Electronics Symposium (IES)*, pp. 214–219, 2019.

[6] M. Riesener, C. Dölle, H. Lauf, and G. Schuh, "Framework for an agile, databased development", *31st CIRP Design Conference 2021*, pp. 343–348, 2021.

[7] M. A. Mascheroni, E. Irrazábal, and G. Rossi, "Continuous Testing Improvement Model", *IEEE/ACM International Conference on Automation of Software Test (AST)*, pp. 109–112, 2021.

[8] P. Zimmerer, "Strategy for Continuous Testing in iDevOps", *IEEE/ACM 40th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pp. 532–533, 2018.

[9] D. Fucci, H. Erdogmus, B. Turhan, M. Oivo, and N. Juristo, "A Dissection of the Test-Driven Development Process: Does It Really Matter to Test-First or to Test-Last?", *IEEE Transactions on Software Engineering*, vol. 43, no. 07, pp. 597–614, 2017.

[10] C. Wang, F. Pastore, A. Goknil, and L. C. Briand, "Automatic Generation of Acceptance Test Cases From Use Case Specifications: An NLP-Based Approach", *IEEE Transactions on Software Engineering*, vol. 48, no. 2, pp. 585–616, 2022.