Início	segunda, 26 de abril de 2021 às 09:04
Estado	Prova submetida
Data de submissão:	segunda, 26 de abril de 2021 às 10:53
Tempo gasto	1 hora 49 minutos
Nota	156,0/200,0
Nota	**15,6** de um máximo de 20,0 (**78**%)

Informação

The iReceptor Plus project is building a technological platform with accompanying bioinformatics and machi facilitate storage, sharing and joint research of specific immunological data, called AIRR-seq data.

This data will serve for basic research as well as developing new therapeutics and vaccines for infectious dise diseases, and cancer. Currently, the consortium is offering full support for anyone who produce COVID-19 re seq data, by assisting to curate, store and share their data using iReceptor Plus platform.

The iReceptor Plus aims to support federated exploration and analysis of distributed repositories of AIRR-sec generating, analyzing, depositing, exploring, and sharing such data.

The high-level architecture includes different types of nodes: web portals, distributed repositories, and large as depicted below.

The High-Level iReceptor Plus Platform Architecture

[just for curiosity, you can find more details at https://www.ireceptor-plus.com]

Pergunta 1

Respondida Pontuou 24,00 de 30,00

Consider the perspective of the overall iReceptor Plus software system and especially the very important chai repositories and computational resources.

In that architecture, which key architectural styles (max. 2-3) do you see as helpful to desig overa

For the explanation please describe the components and connectors that you should exist to impleme styles mentioned.

Firstly we need to have a data-flow architectural style, since the main goal is to explore and analyze data fron repositories. We can also detect a several data transformation steps, that is generating, analyzing, depositing So since these steps are well defined I would choose a Batch Sub-style. In this case, the components are the mentioned beforehand, and the connectors are simply a link between there resources.

Secondly, this data needs to be stored somewhere in order for it to be accessed by several clients, so we can Centered architecture would be a good idea. Since it is important to have distributed repositories then I belie sub-style would be the best option, because we can look at the distributed repositories as Knowledge Source accessed and analyzed by the machine learning tools. Other data such as conclusions from those analysis car knowledge sources that are accesses by clients via the web portal. In this case, the components are clients via machine learning tools, various knowledge sources and central data blackboard.

Pergunta 2

Respondida Pontuou 24,50 de 35,00

iReceptor Plus must support the interoperability of a variety of repository types, possibly differing in the way technologies, capabilities, and provided services.

How to cope with such diversity of repositories to end up with a design that decouples fro[illegible] dive extend and maintain when new types of repositories must be supported? Which design pattern do yo[illegible] candidate?

a. Abstract Factory

b. Adapter

c. Façade

d. Mediator

e. Proxy

Justify.

(i) why the choice for the selection;

(ii) the key benefits and liabilities of using those patterns;

(iii) describing how the pattern's roles can be mapped in the concrete solution.

(i)
In order to support the interoperability of a variety of repository types, then I would choose the adapter patt allow objects with incompatible interfaces to collaborate with each other.

(ii)
One of the advantageous of the adapter pattern is that it allows to separate the interface from the primary lo hand with the Single Responsibility Principle. Another advantage is the ability to introduce several types of ac needing to change the original code, respecting the Open/Closed Principle. The main drawback of using the a large system the code can get complex with the increasing number of adapter classes, so depending on the easier to simply change the main repository classes' codebase so it matches the rest of the system.

(iii)
Firstly we need an interface that will be implemented by all repository types, then we need adapter classes th repository class also implement the repository interface. This will allow for two different repository classes to adapter class, since the adapter class contains a concrete repository it allows for an adaptive communication.

Pergunta 3

Respondida Pontuou 30,00 de 35,00

iReceptor Plus aims to support researchers unaware of AIRR-seq data to use a meta-search engine to discove
their research work. The search engine is one component of iReceptor Plus Gateways, which should be able t
different repositories, interpret and merge the data, and then to produce the expected search res

**How to design the search engine to enable its transparent interaction with all repositories, both to qu
data, and rank, merge and process the results? Which design pattern do you suggest being the best ca**

a. Adapter

b. Interpreter

c. Mediator

d. Strategy

e. Template Method

Justify.

(i) why the choice for the selection;

(ii) the key benefits and liabilities of using those patterns;

(iii) describing how the pattern's roles can be mapped in the concrete solution.

(i)
I think that the best design pattern would be the Strategy Pattern, because it allows us to define a family of ir
algorithms that would help us to deal with the various interactions between different repository types. The te
good option, but since we are looking for interchangeable ways to deal with different repository types, then
the best solution.

(ii)
The strategy pattern allows us to swap algorithms at runtime, this is useful for processing various repository t
let's us introduce new strategies without changing the program's context, this is useful if another repository t
system. Finally, each implementation is isolated and so, easy to maintain. On the other hand, it is important t
have a few repository types, then this approach can over complicate things with new classes and interfaces. A
the client class needs to be aware of which strategy will be used, in order for it to swap strategies, so it must
program's context.

(iii)
In order to implement this pattern, we need a search engine class with a strategy attribute. This strategy is ar
implemented by the different strategies needed in order to process each repository type. Then when a client
repository, the search engine class should be aware of which type of repository is being queried in order to c
strategy.

Pergunta 4

Respondida Pontuou 17,50 de 35,00

AIRR-seq data is complex, rich and also very sensitive, since it contains a huge volume of information about a preserved private, at all means. Basic statistical analysis of features or complex analyses within that data vary computational complexity and privacy sensitivity. Although iReceptor Plus Gateways have differer es of a is common that researchers want to define their own ways of analyzing the data.

Which design patterns do you suggest implementing in the gateway to enable the gateway to run ana researcher but without requiring the data to be disclosed and exposed to the researcher?

a. Command

b. Interpreter

c. Iterator

d. Strategy

e. Visitor

Justify.

(i) why the choice for the selection;

(ii) the key benefits and liabilities of using those patterns;

(iii) describing how the pattern's roles can be mapped in the concrete solution.

(i)
I think that the Interpreter and Iterator Pattern are a good choice, because it allows us to interpret the high-le representation provided by the Researcher and also define a series of algorithms to transverse a collection of the collection itself.

(ii)
The interpreter pattern is used to interpret a high-level program representation in order to transform it into s readable. The main problem for this pattern is that it involves a high investment that may not pay off in the lo pattern let's us clean up the code, making it simpler for external use. It also allows us to introduce new featur changes in the base class (Open/Closed principle). However, applying this pattern might be too much effort i iterating over are not that complex, on the other hand iterating over a complex data structure using an iterat performant than going thought the collection by hand.

(iii)
The interpreter should be able to process the analysis and build an algorithm accordingly, secondly this algor too access data in order for the data to remain unexposed to the researcher.

Pergunta 5

Respondida Pontuou 35,00 de 35,00

Technically speaking, some optimal advanced processing and storage services would recommend that some one computational node to others. But such transfers may not be possible, for node reasons, such as legal, et technical policies (e.g. GDPR, national data protection, data volume, computational power, netwo dwidt

How do you suggest coping with the management of such complex set of policies of each node and h nodes and overall system?

a. Chain of Responsibility

b. Iterator

c. Mediator

d. Observer

e. Strategy

Justify.

(i) why the choice for the selection;

(ii) the key benefits and liabilities of using those patterns;

(iii) describing how the pattern's roles can be mapped in the concrete solution.

(i)
I would choose the chain of responsibility design pattern. This pattern allows us to pass along requests along this case nodes. Upon receiving these requests, each handler decides either to process that request or to pas handler in the chain. This is applicable to this problem, because if a node is requested to do something that i it may pass along this request along the nodes until it reaches on that has permission to handle the request.

(ii)
By using the Chain of Responsibility pattern, we can decouple classes that invoke operations from classes tha operations, obeying the Single Responsibility Pattern. Also, we can easily add new handlers if the need arises requests without breaking the existing code, Open/Closed Principle. On the other hand, this design pattern d requests are handled, as we have no way of knowing if there is a handler in the chain able to process this req

(iii)
We must define a handler interface that will be implemented by the node class, each node has access to the Upon receiving a request, the node decides if it can handle that request, if so it will process it, otherwise it wi node in the chain.

Pergunta 6

Respondida Pontuou 25,00 de 30,00

AIRR-seq data is usually very large. The larger the data set, the larger amount of storage and processing capa for the system, requiring a well-defined strategy for scalability.

Identify in the system architecture the likely bottlenecks and suggest strategies to mitigate . Iden the practices or patterns adopted, describing how they can be instantiated with technology.

In a large and complex system, the back office might become too complex to maintain over time, so a good would be the Façade pattern in order to simplify internal usage.

Since we are talking about a web portal, the data-centered architecture may cause the database to be floode might cause performance problems. Hence, by using the proxy design pattern, we are able to control which r what time. This alongside with the Memento design pattern allows us to cache the results of some requests, respond to repeated incoming requests.

◄ MIEIC-ASSO-past-tests-bundle

Ir para...