Aufträge Technologierecherche

1 JavaFX mit MVVM vs. MVC vs. ???

- Gibt es neben MVVM und MVC vielleicht andere Architekturmuster die sich besser eignen?
- Welche Architektur ist einfacher/besser kompatibel mit JavaFX?
- Ggf. hilfreich: https://github.com/sialcasa/mvvmFX/wiki
- Erstellen eines Mini-Beispielprojektes mit der gewählten Architektur und JavaFX
- Upload im git Projekt im Ordner architecture/research

Interessant für: Jan Liebeck

2 JavaFX UI Erzeugungstool: JavaFX Scene Builder vs. Gluon vs. ???

- Ähnlich wie bei der Android-App entwicklung werden in JavaFX die GUIs mit einem XML Format (FXML) beschrieben. Das muss man aber nicht komplett von Hand machen, sondern es gibt Tools über die das grafisch geht
- Es soll evaluiert werden welches Tool (JavaFX Scene Builder vs. Gluon vs. ???) für uns gut geeignet ist
- Erstellen eines Mini-Beispielprojektes unter Verwendung des gewählten Tools
- Upload im git Projekt im Ordner architecture/research

Interessant für: Leon Kalb, Robin Kayser (2)

3 JavaFX UI Design: Realisieren von moderner Erscheinung

- Java Anwendungen sehen leider oft so aus als hätten sie den Wechsel ins 21. Jahrhundert nicht erlebt. Welche Libraries können wir nutzen und worauf müssen wir achten, damit unsere GUI einigermaßen modern aussieht?
- Der Post hier könnte hilfreich sein: https://medium.com/@keeptoo/javafx-java-modern-ui-design-starter-pack-aab1c331fd3c
- Erstellen eines Mini-Beispielprojektes welches “moderne” Komponenten verwendet
- Upload im git Projekt im Ordner architecture/research

Interessant für: Jan Liebeck, Benjamin Killisch(3), Jan Eric Schulze(2), Robin Kayser (1)

4 JavaFX Best Practices

- Welche Best-Practices gibt es?
- Worauf sollten wir achten um gute JavaFX Anwendungen zu schreiben? Gut heißt soviel wie modularisierbar, performant, wartbar, erweiterbar, folgt den SOLID Prinzipien
- Evlt. hilfreich: https://docs.oracle.com/javafx/2/best_practices/jfxpub-best_practices.htm
- Erstellen eines kleinen Cheatsheet, welches die wichtigsten Best-Pracices veranschaulicht
- Upload im git Projekt im Ordner architecture/research

  Interessant für: Jan Eric Schulze (1), Elias Anton

5 Programming Workflow Evaluation

- Zur Steigerung der Code-Qualität sollte der Code unabhängig vom jeweiligen Programmierer unseren Coding-Guidelines (TBD) folgen und getestet sein. Dafür sollen zwei Punkte realisiert werden:
  - Unit Tests mit JUnit 5 und Mockito werden verwendet
  - Gradle Build schlägt fehl bei Missachtung der Coding Guidelines
- Dazu soll ein Mini-Projekt erstellt werden, an dem die Grundlagen von Junit 5 und Mockito nachvollziehbar vorgestellt werden und bei dem der Gradle-build nur erfolgreich ist wenn ein Tool wie Checkstyle , SpotBugs oder pmd den Code überprüft hat und keine Fehler geworfen hat
- Upload im git Projekt im Ordner architecture/research

  Interessant für: Paul Beck (1), Benjamin Killisch(2)

6 Best Practices zur Erstellung einer Library in Java

- Um unseren Code besser zu modularisieren verwenden wir ein Multimodulprojekt. Das bedeutet er wird in eine GUI Anwendung mit JavaFX und eine Evolutionäre Algorithmen Library geteilt. Hier geht es nun darum worauf man beim Erstellen einer Java-Library achten sollte
- Evtl. hilfreich:
    - https://docs.gradle.org/current/samples/sample_building_java_applications_multi_project.html
    - https://www.baeldung.com/design-a-user-friendly-java-library
    - https://www.oracle.com/corporate/features/library-in-java-best-practices.html
- Erstellen eines kleinen Cheatsheet, welches die wichtigsten Best-Practices veranschaulicht
- Upload im git Projekt im Ordner architecture/research

  Interessant für: Nicas Frank(2), Paul Beck (2), Elias Anton

7 Evaluation der Jenetics Library für unsere Anforderungen

Herrn Weickers Kommentar
Aus meiner Erfahrung heraus würde ich von der Nutzung von EA/GA-Frameworks abraten, da diese hier für Ihre Aufgabe das Leben nicht einfacher machen. 1. Wie sind mit der Beschränkung auf Binärstrings als Genotyp stark eingeschränkt, sodass die Komplexität von Jenetics die Programmierung eher komplizierter macht. Alles, was wirklich GA-typisch ist, ist selbst in wenigen Zeilen programmiert! 2. Der Ansatz, Abläufe beliebig zusammenklickbar zu machen, ist schon sehr speziell, da ich dort im nächsten Semester auf koevolutionäre Ansätze, Meta-GAs, etc abziele. Dort werden vermutlich die Grenzen der Jenetics-Bibliothek erreicht, wodurch die Verwendung der Library eher zum Hindernis wird. Aus Kundensicht würde ich eine schlanke und spezifische Implementierung bevorzugen.

- Jenetics ist eine umfangreiche Bibliothek zur Arbeit mit evolutionären Algorithmen. Wenn wir sie verwenden können sparen wir uns sehr viel Entwicklungsaufwand
- Wir können sie allerdings nur dann verwenden, wenn sie unseren Anforderungen entspricht. Heißt konkret folgendes muss möglich sein:

    - Nach einem Optimierungslauf des evolutionären Algorithmus können Statistiken und Visualisierungen für die einzelnen Datenspeicher über den Verlauf der Optimierung angezeigt werden.
    - In der GUI werden die Algorithmen anhand des Pipes & Filters Konzeptes realisiert. Kann die Library in diesem Szenario verwendet werden?
- Sicherstellen, dass die beiden Anforderungen umsetzbar sind und erstellen eines Cheatsheet, welches die Konfigurationsmöglichkeiten von Jenetics aufzeigt zeigt
- Upload im git Projekt im Ordner architecture/research

Interessant für: Nicas Frank(1), Benjamin Killisch(1)

8 Architektur/Struktur der Library

- Welche Architekturschema gibt es und eignen sich für die Library? Gesucht ist eine Ausarbeitung mit einem Vergleich von Vor- und Nachteilen
- Achtung: Ein häufiger Denkfehler ist, dass die Architektur-Schichten durch die Package Struktur abgebildet werden. Meist ist es jedoch besser die Packages nach Feature zu strukturieren. Siehe: https://phauer.com/2020/package-by-feature/
- Ein Paar Denkanstöße
    - Clean Architecture
    - Onion Architecture
    - Schichten Architektur
- Evtl. hilfreich
    - https://softwareengineering.stackexchange.com/questions/371966/is-clean-architecture-by-bob-martin-a-rule-of-thumb-for-all-architectures-or-i

Interessant für: Jan Liebeck, Paul Beck (3)