INTERNATIONALE HOCHSCHULE

PORTFOLIO

Aufgabenstellung zum Kurs: Projekt: Data Engineering (DLMDWWDE02)

INHALTSVERZEICHNIS

1. AUFGABENSTELLUNG

Sowohl das Datenvolumen als auch die Verfügbarkeit von Daten wachsen rapide, was es uns erlaubt, aufwendigere Applikationen auf Basis dieser Daten zu erstellen, die informativere Ergebnisse produzieren. Auf der einen Seite kann das zu produktiveren Arbeitsprozessen und einem besseren Verständnis von Phänomenen und Systemen führen. Auf der anderen Seite sehen sich Data Engineers mit der Herausforderung konfrontiert, immer größere Datenmengen in kurzer Zeit verarbeiten zu müssen. Da außerdem datengetriebene Prozesse für die meisten Organisationen einen immer zentraleren Stellenwert annehmen, wächst die Anforderungen an datenintensive Applikationen im Hinblick auf Robustheit, Verlässlichkeit (*reliability*), Skalierbarkeit (*scalability*) und Wartbarkeit (*maintainability*).

In diesem Portfolio Projekt wirst Du eine Datenarchitektur für eine datenintensive großskalige Applikation entwickeln. Dabei wirst Du verschiedene Techniken, wie beispielsweise *Infrastruktur as Code (IaC)* oder *Microservices* anwenden. Am Ende dieses Projekts wirst Du bewiesen haben, dass Du über praktische Fähigkeiten verfügst, die es Dir ermöglichen das Backend für eine datenintensive Applikation zu implementieren. Dabei wirst Du Dich in eine oder mehrere State-of-the-Art-Technologien vertiefen und diese praktisch anwenden.

Im Rahmen dieses Projektes entscheidest Du Dich für eine der beiden folgenden Aufgaben. Angesichts weit verbreiteter Architekturen und Standards zur Datenprozessierung ähneln sich beide Aufgaben, unterscheiden sich allerdings in der Ausrichtung der Anforderungen. Während die erste Aufgabe Batch-Prozessierungen vorsieht, sind die Anforderungen in der zweiten Aufgabe auf Stream-Prozessierung ausgelegt.

1.1. Aufgabe 1: Entwickle eine batch-basierte Datenarchitektur für eine datenintensive Applikation

In dieser Aufgabe entwickelst und implementierst Du die zugrundeliegende Dateninfrastruktur als Backend für eine datenintensive *Machine Learning* Applikation. Du entwickelst ein System, was in der Lage ist, massive Datenmengen aufzunehmen, diese auf effiziente Weise zu speichern, diese Daten zu prozessieren, zu aggregieren, und für die direkte Nutzung in einer *Machine Learning* Applikation zur Verfügung zu stellen (diese *Machine Learning* Applikation ist nicht Teil dieses Projektes). Die Frontend Applikation wird einmal pro Quartal ausgeführt und erzeugt eine neue Version eines *Machine Learning* Modells auf Basis der neuen Daten in der Dateninfrastruktur. Folglich entwickelst Du eine Dateninfrastruktur, die in der Lage ist, Datenprozessierungen in Batches durchzuführen. Auf Deiner Reise durch State-of-the-Art-Architekturen zur Datenprozessierung wirst Du lernen, übliche Data Engineering Prinzipien anzuwenden. Außerdem wirst Du Dich eigenständig in eine oder mehrere verbreitete Softwarekomponenten vertiefen.

Die Maßnahme: Entwickle und implementiere ein Datensystem zur Batch-Prozessierung, welches verlässlich (*reliable*), skalierbar (*scalable*) und wartbare (*maintainable*) ist.

Die Implementierung erfolgt in drei Schritten:

1.1.1. Konzeptionsphase

Dieser Teil ist der Wichtigste für Dein Portfolio Projekt. Alles, was in dieser Phase übersehen oder vergessen wird, hat einen negativen Einfluss auf die spätere Implementierung und könnte im schlimmsten Fall zu nutzlosen Ergebnissen führen.

Der erste Schritt in Deinem Portfolio Projekt besteht darin, sich einen ersten **Überblick über verfügbare weitverbreitete Softwarekomponenten** zu verschaffen, die häufig in datenintensiven Applikationen verwendet werden, wie beispielsweise Apache Kafka, das Hadoop Ökosystem oder Spark. Du musst Dich nicht in jedes Detail für jede verfügbares Software vertiefen, aber Du solltest in der Lage sein, die zur Verfügung stehenden weit verbreiteten Softwarekomponenten mit einer bestimmten Aufgabe in einer Dateninfrastruktur in Verbindung zu bringen. Außerdem solltest Du Dich in dieser Phase damit beschäftigen, welche Technologien geeignet sind, um die **Verfügbarkeit (*reliability*), Skalierbarkeit (*scalability*) und Wartbarkeit (*maintainability*)** Deines Systems sicherzustellen. Du solltest Dir auch jetzt schon Gedanken darüber machen, wie die Aspekte **Datensicherheit, Datenschutz und Data Governance** in Deinem Datensystem berücksichtigt werden können.

Ein weiterer Aspekt, über den Du Dir in dieser Phase Gedanken machen solltest, ist, dass Dein System von Anfang an reproduzierbar sein sollte. Es macht also Sinn, eine Lösung zur **Versionskontrolle**, wie beispielsweise GitHub, und/oder Techniken wie **Infrastructure as Code (IaC)** zu verwenden.

Um Dein System sicher und nach Ausfällen wiederherstellbar zu machen, entwickelst Du Dein Datensystem auf eine Weise, dass einzelne Komponenten im Sinne einer **Microservice Architektur** voneinander isoliert und unabhängig sind. Um das zu erreichen, solltest Du Dich mit den Prinzipien der **Containerisierung** vertraut machen und Dir Basiskenntnisse in einer entsprechenden Technologie, beispielsweise in Docker, aneignen.

Als Hilfestellung können Dir **Best-Practice-Architekturen** dienen, welche Du beispielsweise in den Dokumentationen der großen Cloud Provider, aber auch in zahlreichen Blogposts und ähnlichen online-Formaten finden kannst.

Nachdem Du Dir eigenständig einen Überblick über diese weit verbreiteten Technologien verschafft hast, wählst Du als nächstes eine **Datenquelle** für Dein Projekt. Ein guter Startpunkt für diese Suche ist beispielsweise Kaggle, wo Du eine Vielzahl offener Beispieldatensätze findest. Da es in diesem Projekt nicht um den Inhalt der Anwendung, sondern vielmehr um die Methodik zum Aufbau einer Dateninfrastruktur geht, ist es zweitrangig, was die von Dir gewählten Daten beschreiben. Die einzige Anforderung an die gewählte Datenquelle ist, dass es sich um ein großes Datenvolumen handeln sollte (mindestens 1,000,000 Datenpunkte) und dass es für jeden Datenpunkt einen Zeitstempel gegeben sollte.

Am Ende dieser Phase fertigst Du eine **konzeptionelle Zeichnung Deiner Dateninfrastruktur** an. Außerdem solltest Du dabei für Dich selbst folgende Fragen beantworten können:

- Welcher Microservice wird *data ingestion* in mein System übernehmen?
- Welcher Microservice wird die Datenspeicherung in meinem System übernehmen?
- Welcher Microservice wird die Datenvorprozessierung und Aggregation übernehmen?
- Mit welchen Techniken und Methoden stelle ich sicher, dass mein System zuverlässig (*reliable*), skalierbar (*scalable*) und wartbar (*maintainable*) ist?
- Mit welchen Techniken stelle ich sicher, dass Datenschutz, Datensicherheit und Data Governance in meinem System berücksichtigt werden?
- Welche *Docker Images* werde ich nutzen, um mein System aufzubauen und müssen diese gegebenenfalls modifiziert werden?
- Welche Daten werde ich verwenden, um mein System zu testen?
- Mit welcher zeitlichen Frequenz werden Daten in mein System geladen, prozessiert, aggregiert und zur Verfügung gestellt?

Für jeden dieser Punkte solltest Du In der Lage sein, eine angemessene Begründung zu geben. Diese Punkte sind für Dich gedacht, um Deine Arbeit zu strukturieren. Diese Fragen müssen nicht wörtlich in Deiner Abgabe beantwortet werden.

Abschließend stellst Du Deine Gedanken und Recherchen in einer **konzeptionellen Flow-Chart-Zeichnung zum Entwurf Deiner Architektur** zusammen. Zur Inspiration kannst Du beispielsweise die Referenzarchitekturen der großen Cloud Provider nutzen (bspw. Microsoft Azure Reference Architecture: https://docs.microsoft.com/en-gb/azure/architecture/browse/ [letzter Zugriff: 11.11.2020]). Außerdem solltest Du die **Vor- und Nachteile** Deiner konzeptionellen Architektur diskutieren.

Während des gesamten Prozesses gibt es im Rahmen der Online-Tutorien die Möglichkeit, über Ideen und Entwürfe zu sprechen und sich Feedback einzuholen. In den Online-Tutorien werden exemplarisch Arbeiten besprochen, die der Tutorin/dem Tutor zuvor übermittelt wurden. Hier besteht für alle die Möglichkeit, sich einzubringen und vom Feedback der anderen zu lernen. **Es wird empfohlen, diese Kanäle zu nutzen, um Fehler zu vermeiden und Verbesserungen vorzunehmen. Erst danach sollen die Ergebnisse in der ersten Phase zur Bewertung abgegeben werden.** Hier erfolgt ein abschließendes Feedback Durch die Tutorin/den Tutor und die Arbeit in der zweiten Phase kann beginnen.

1.1.2. Erarbeitungs-/Reflexionsphase

In dieser Phase findet die praktische Implementierung Deines Datensystems statt. Du setzt ein **Git Repository** auf, in welchem Du den gesamten Code zur Verfügung stellst, den Du während Deines Projektes generierst. Du implementierst Deine Dateninfrastruktur als Microservice Architektur entsprechend der konzeptionellen Visualisierung, die Du in der Konzeptionsphase angefertigt hast. Zur Implementierung nutzt Du **Docker Container**, die Du gegebenenfalls anpassen musst, damit sie den Anforderungen Deines Systems entsprechen. Dabei sollte besonderer Wert auf **Verlässlichkeit (*reliability*), Skalierbarkeit (*scalability*), Wartbarkeit (*maintainability*), Datensicherheit (*data security*), Data Governance, und Datenschutz (*data protection*)** gelegt werden. Im Rahmen dieses Projektes deployst Du Dein System auf Deiner lokalen Maschine und es ist nicht notwendig ein Deployment in der Cloud vorzunehmen. Nachdem alle Deine Microservices laufen und miteinander kommunizieren, **lädst Du die Beispieldaten** in Dein System. Du solltest sicherstellen, dass die Daten wie erwartet geladen, prozessiert und aggregiert werden. Am Ende dieser Phase hast Du eine Arbeitsumgebung geschaffen, die **reproduzierbar** ist und welche genutzt werden kann, um eine Dateninfrastruktur aufzubauen. Dein Code befindet sich in einem versionskontrollierten Git Repository, welches in sich ein eigenes Portfolio darstellt und welches Du zukünftigen potenziellen Arbeitgebern präsentieren kannst.

Während des gesamten Prozesses gibt es im Rahmen der Online-Tutorien und der weiteren Kanäle die Möglichkeit, über Ideen und Entwürfe zu sprechen und sich ausreichend Rückmeldung, Tipps und Hinweise zu holen. **Es wird empfohlen, diese Kanäle zu nutzen, um Fehler zu vermeiden und Verbesserungen vorzunehmen. Erst danach sollen die Ergebnisse in der zweiten Phase zur Bewertung abgegeben werden**. Nach dem folgenden abschließenden Feedback Durch die Tutorin/den Tutor wird in der dritten Phase an dem finalen Entwurf weitergearbeitet.

1.1.3. Finalisierungsphase

In dieser Phase schließt Du Dein Projekt ab. Zunächst nimmst Du ein Finetuning an Deinem Code vor. Geh erneut Durch jeden einzelnen Prozess und überprüfe an welcher Stelle Dinge flüssig liefen und wo Probleme auftauchten. Als Orientierung können Dir folgende Fragen dienen, welche es Dir erleichtern können, Dein Projekt konstruktiv abzuschließen:

- Werden alle technischen Voraussetzungen von meinem System erfüllt?
- Was lief schief und warum?
- Ist mein System zuverlässig (*reliable*), skalierbar (*scalable*) und leicht zu warten (*maintainable*)?
- Welche Maßnahmen können ergänzt werden, um beispielsweise die Datensicherheit, den Datenschutz oder Data Governance zu verbessern?
- Was werde ich in einem zukünftigen Projekt anders machen, um den Arbeitsprozess zu verbessern?
- Was sind die Hauptschritte, die ich unternommen habe, damit mein Projekt ein Erfolg wird?
- Was sind die drei wertvollsten technischen Skills, die ich während dieses Projektes gelernt habe?
- Was sind die drei wertvollsten Soft Skills, die ich während meines Projektes gelernt habe?

Abschließend solltest Du darüber diskutieren, welche Strategien verfolgt werden können, um eine zweite Daten-Pipeline in Dein System zu integrieren, welche eine Stream-Prozessierung der Daten ermöglicht.

Auch in der Finalisierungsphase gibt es im Rahmen der Online-Tutorien und der weiteren Kanäle die Möglichkeit, sich ausreichend Rückmeldung, Tipps und Hinweise zu holen, **bevor** das fertige Produkt final abgegeben wird. **Es wird empfohlen, diese Kanäle zu nutzen, um Fehler zu vermeiden und Verbesserungen vorzunehmen**. Das **fertige Produkt** wird mit den **Ergebnissen aus Phase 1 und Phase 2** sowie zusammen mit den **oben genannten Materialien** eingereicht. Gewünscht ist zusätzlich **ein Abstract,** das die Lösung der Aufgabe inhaltlich und konzeptionell beschreibt und einen **kurzen Breakdown (making of)** über die technische Herangehensweise nüchtern und informativ darlegt. Das Einfügen eines zusätzlichen zip-Ordners ist hier nicht nötig.

1.2. Aufgabe 2: Entwickle ein real-time Backend für eine datenintensive Applikation

In dieser Aufgabe entwickelst und implementierst Du die zugrundeliegende Dateninfrastruktur als Backend für ein real-time Reporting. Du entwickelst ein System, was in der Lage ist, kontinuierlich massive Datenmengen aufzunehmen, diese auf effiziente Weise zu speichern, zu prozessieren, zu aggregieren, und für die direkte Nutzung in einer Echtzeit-Reporting Applikation zur Verfügung zu stellen (das Reporting ist nicht Teil dieses Projektes). Auf Deiner Reise durch State-of-the-Art-Architekturen zur Echtzeit-Datenprozessierung wirst Du lernen, übliche Data Engineering Prinzipien anzuwenden. Außerdem wirst Du Dich eigenständig in eine oder mehrere verbreitete Softwarekomponenten vertiefen.

Die Maßnahme: Entwickle und implementiere ein Datensystem zur Batch-Prozessierung, welches verlässlich (reliable), skalierbar (scalable) und wartbare (maintainable) ist.

Die Implementierung erfolgt in drei Schritten:

1.2.1. Konzeptionsphase

Dieser Teil ist der Wichtigste für Dein Portfolio Projekt. Alles, was in dieser Phase übersehen oder vergessen wird, hat einen negativen Einfluss auf die spätere Implementierung und könnte im schlimmsten Fall zu nutzlosen Ergebnissen führen.

Der erste Schritt in Deinem Portfolio Projekt besteht darin, sich einen ersten **Überblick über verfügbare weitverbreitete Softwarekomponenten** zu verschaffen, die häufig in datenintensiven Echtzeit-Applikationen verwendet werden, wie beispielsweise Apache Kafka oder Spark Streaming. Du musst Dich nicht in jedem Detail für jede verfügbare Software vertiefen, aber Du solltest in der Lage sein, die zur Verfügung stehenden weit verbreiteten Softwarekomponenten mit einer bestimmten Aufgabe in einer Dateninfrastruktur in Verbindung zu bringen. Außerdem solltest Du Dich in dieser Phase damit beschäftigen, welche Technologien geeignet sind, um die **Verfügbarkeit (*reliability*), Skalierbarkeit (*scalability*) und Wartbarkeit (*maintainability*)** Deines Systems sicherzustellen. Du solltest Dir auch jetzt schon Gedanken darüber machen, wie die Aspekte **Datensicherheit, Datenschutz und Data Governance** in Deinem Datensystem berücksichtigt werden können.

Ein weiterer Aspekt, über den Du Dir in dieser Phase Gedanken machen solltest, ist, dass Dein System von Anfang an reproduzierbar sein sollte. Es macht also Sinn, eine Lösung zur **Versionskontrolle**, wie beispielsweise GitHub, und/oder Techniken wie **Infrastructure as Code (IaC)** zu verwenden.

Um Dein System sicher und nach Ausfällen wiederherstellbar zu machen, entwickelst Du Dein Datensystem auf eine Weise, dass einzelne Komponenten im Sinne einer **Microservice Architektur** voneinander isoliert und unabhängig sind. Um das zu erreichen, solltest Du Dich mit den Prinzipien der **Containerisierung** vertraut machen und Dir Basiskenntnisse in einer entsprechenden Technologie, beispielsweise in Docker, aneignen.

Als Hilfestellung können Dir **Best-Practice-Architekturen** dienen, welche Du beispielsweise in den Dokumentationen der großen Cloud Provider, aber auch in zahlreichen Blogposts und ähnlichen online-Formaten finden kannst.

Nachdem Du Dir eigenständig einen Überblick über diese weit verbreiteten Technologien verschafft hast, wählst Du als nächstes eine **Datenquelle** für Dein Projekt. Ein guter Startpunkt für diese Suche ist beispielsweise Kaggle, wo Du eine Vielzahl offener Beispieldatensätze findest. Da es in diesem Projekt nicht um den Inhalt der Anwendung, sondern vielmehr um die Methodik zum Aufbau einer Dateninfrastruktur geht, ist es zweitrangig, was die von Dir gewählten Daten beschreiben. Die einzige Anforderung an die gewählte Datenquelle ist, dass es sich um ein großes Datenvolumen handeln sollte (mindestens 1,000,000 Datenpunkte) und dass es für jeden Datenpunkt einen Zeitstempel gegeben sollte. Da es sich in dieser Aufgabe um eine Echtzeit-Applikation handelt, solltest Du

nach offenen APIs für Streaming-Daten suchen. Alternativ kannst Du eine kleine Anwendung auf Deinem lokalen Rechner ausführen, welche kontinuierlich Daten produziert, indem bspw. Temperaturdaten simuliert werden. Eine dritte Möglichkeit besteht darin, einen statischen Datensatz zu verwenden, in dem Änderungen augenblicklich Auswirkungen in Deinem System zur Folge haben.

Am Ende dieser Phase fertigst Du eine **konzeptionelle Zeichnung Deiner Dateninfrastruktur** an. Außerdem solltest Du dabei für Dich selbst folgende Fragen beantworten können:

- Welcher Microservice wird *data ingestion* in mein System übernehmen?
- Welcher Microservice wird die Datenvorprozessierung und Aggregation übernehmen?
- Mit welchen Techniken und Methoden stelle ich sicher, dass mein System zuverlässig (*reliable*), skalierbar (*scalable*) und wartbar (*maintainable*) ist?
- Mit welchen Techniken stelle ich sicher, dass Datenschutz, Datensicherheit und Data Governance in meinem System berücksichtigt werden?
- Welche *Docker Images* werde ich nutzen, um mein System aufzubauen und müssen diese gegebenenfalls modifiziert werden?
- Welche Daten werde ich verwenden, um mein System zu testen?
- Welche Aggregations- und Windowing-Funktionen werde ich nutzen, um die Daten zu aggregieren?

Für jeden dieser Punkte solltest Du In der Lage sein, eine angemessene Begründung zu geben. Diese Punkte sind für Dich gedacht, um Deine Arbeit zu strukturieren. Diese Fragen müssen nicht wörtlich in Deiner Abgabe beantwortet werden.

Abschließend stellst Du Deine Gedanken und Recherchen in einer **konzeptionellen Flow-Chart-Zeichnung zum Entwurf Deiner Architektur** zusammen. Zur Inspiration kannst Du beispielsweise die Referenzarchitekturen der großen Cloud Provider nutzen (bspw. Microsoft Azure Reference Architecture: https://docs.microsoft.com/en-gb/azure/architecture/browse/ [letzter Zugriff: 11.11.2020]). Außerdem solltest Du die **Vor- und Nachteile** Deiner konzeptionellen Architektur diskutieren.

Während des gesamten Prozesses gibt es im Rahmen der Online-Tutorien die Möglichkeit, über Ideen und Entwürfe zu sprechen und sich Feedback einzuholen. In den Online-Tutorien werden exemplarisch Arbeiten besprochen, die der Tutorin/dem Tutor zuvor übermittelt wurden. Hier besteht für alle die Möglichkeit, sich einzubringen und vom Feedback der anderen zu lernen. **Es wird empfohlen, diese Kanäle zu nutzen, um Fehler zu vermeiden und Verbesserungen vorzunehmen. Erst danach sollen die Ergebnisse in der ersten Phase zur Bewertung abgegeben werden.** Hier erfolgt ein abschließendes Feedback Durch die Tutorin/den Tutor und die Arbeit in der zweiten Phase kann beginnen.

1.2.2. Erarbeitungs-/Reflexionsphase

In dieser Phase findet die praktische Implementierung Deines Datensystems statt. Du setzt ein **Git Repository** auf, in welchem Du den gesamten Code zur Verfügung stellst, den Du während Deines Projektes generierst. Du implementierst Deine Dateninfrastruktur als Microservice Architektur entsprechend der konzeptionellen Visualisierung, die Du in der Konzeptionsphase angefertigt hast. Zur Implementierung nutzt Du **Docker Container**, die Du gegebenenfalls anpassen musst, damit sie den Anforderungen Deines Systems entsprechen. Dabei sollte besonderer Wert auf **Verlässlichkeit (*reliability*), Skalierbarkeit (*scalability*), Wartbarkeit (*maintainability*), Datensicherheit (*data security*), Data Governance, und Datenschutz (*data protection*)** gelegt werden. Im Rahmen dieses Projektes deployst Du Dein System auf Deiner lokalen Maschine und es ist nicht notwendig ein Deployment in der Cloud vorzunehmen. Nachdem alle Deine Microservices laufen und miteinander kommunizieren,

lädst Du die Beispieldaten in Dein System. Du solltest sicherstellen, dass die Daten wie erwartet geladen, prozessiert und aggregiert werden. Am Ende dieser Phase hast Du eine Arbeitsumgebung geschaffen, die **reproduzierbar** ist und welche genutzt werden kann, um eine Dateninfrastruktur aufzubauen. Dein Code befindet sich in einem versionskontrollierten Git Repository, welches in sich ein eigenes Portfolio darstellt und welches Du zukünftigen potenziellen Arbeitgebern präsentieren kannst.

Während des gesamten Prozesses gibt es im Rahmen der Online-Tutorien und der weiteren Kanäle die Möglichkeit, über Ideen und Entwürfe zu sprechen und sich ausreichend Rückmeldung, Tipps und Hinweise zu holen. **Es wird empfohlen, diese Kanäle zu nutzen, um Fehler zu vermeiden und Verbesserungen vorzunehmen. Erst danach sollen die Ergebnisse in der zweiten Phase zur Bewertung abgegeben werden**. Nach dem folgenden abschließenden Feedback Durch die Tutorin/den Tutor wird in der dritten Phase an dem finalen Entwurf weitergearbeitet.

1.2.3. Finalisierungsphase

In dieser Phase schließt Du Dein Projekt ab. Zunächst nimmst Du ein Finetuning an Deinem Code vor. Geh erneut Durch jeden einzelnen Prozess und überprüfe an welcher Stelle Dinge flüssig liefen und wo Probleme auftauchten. Als Orientierung können Dir folgende Fragen dienen, welche es Dir erleichtern können, Dein Projekt konstruktiv abzuschließen:

- Werden alle technischen Voraussetzungen von meinem System erfüllt?
- Was lief schief und warum?
- Ist mein System zuverlässig (*reliable*), skalierbar (*scalable*) und leicht zu warten (*maintainable*)?
- Welche Maßnahmen können ergänzt werden, um beispielsweise die Datensicherheit, den Datenschutz oder Data Governance zu verbessern?
- Was werde ich in einem zukünftigen Projekt anders machen, um den Arbeitsprozess zu verbessern?
- Was sind die Hauptschritte, die ich unternommen habe, damit mein Projekt ein Erfolg wird?
- Was sind die drei wertvollsten technischen Skills, die ich während dieses Projektes gelernt habe?
- Was sind die drei wertvollsten Soft Skills, die ich während meines Projektes gelernt habe?

Abschließend solltest Du darüber diskutieren, welche Strategien verfolgt werden können, um eine zweite Daten-Pipeline in Dein System zu integrieren, welche eine Barch-Prozessierung der Daten ermöglicht.

Auch in der Finalisierungsphase gibt es im Rahmen der Online-Tutorien und der weiteren Kanäle die Möglichkeit, sich ausreichend Rückmeldung, Tipps und Hinweise zu holen, **bevor** das fertige Produkt final abgegeben wird. **Es wird empfohlen, diese Kanäle zu nutzen, um Fehler zu vermeiden und Verbesserungen vorzunehmen**. Das **fertige Produkt** wird mit den **Ergebnissen aus Phase 1 und Phase 2** sowie zusammen mit den **oben genannten Materialien** eingereicht. Gewünscht ist zusätzlich **ein Abstract,** das die Lösung der Aufgabe inhaltlich und konzeptionell beschreibt und einen **kurzen Breakdown (making of)** über die technische Herangehensweise nüchtern und informativ darlegt. Das Einfügen eines zusätzlichen zip-Ordners ist hier nicht nötig.

2. BETREUUNGSPROZESS

Bei der Betreuung der Portfolios stehen grundsätzlich mehrere Kanäle offen. Die jeweilige Inanspruchnahme liegt dabei im eigenen Verantwortungsbereich. Die eigenständige Erarbeitung eines Produktes und die Befüllung der jeweiligen Portfolioteile ist dabei Teil der zu erbringenden Prüfungsleistung und fließt in die Gesamtbewertung mit ein.

Zum einen sieht die tutorielle Betreuung Feedbackschleifen zu den einzureichenden Portfolioteilen im Rahmen der Konzeptions- sowie der Erarbeitungs- und Reflexionsphase vor. Das Feedback erfolgt im Rahmen einer Einreichung des jeweiligen Portfolioteils. Des Weiteren werden regelmäßige Online-Tutorien angeboten, in denen Gelegenheit besteht, mit der Tutorin/dem Tutor Fragen zur Bearbeitung des Portfolios zu besprechen. Die Tutorin/der Tutor steht zusätzlich für fachliche Rücksprachen sowie für formale und allgemeine Fragen zum Vorgehen bei der Portfoliobearbeitung zur Verfügung.

Technische Fragen zur Nutzung von PebblePad sind per Mail an das Prüfungsamt zu richten.

3. ZUSATZINFORMATIONEN ZUR BEWERTUNG

In die Bewertung des Portfolios fließen die folgenden Kriterien mit dem jeweils angegebenen Prozentsatz ein:

Bewertungskriterien	Erläuterungen	Gewichtung
Problemabgrenzung/Zielsetzung	*Erfassung des Problems *Klare Problemabgrenzung/Zielsetzung *Nachvollziehbares Konzept	10%
Methodik/Idee/Vorgehen	*Angemessener Transfer von Theorien/Modellen *Klare Angaben zur gewählten Methodik/zur gewählten Idee/zum gewählten Vorgehen	20%
Qualität der Umsetzung	*Qualität der Umsetzung und Dokumentation	40%
Kreativität/Richtigkeit	*Kreativität des Lösungsansatzes *Umgesetzte Lösung erfüllt angestrebte Zielsetzung	20%
Formale Anforderungen	*Einhaltung der formalen Vorgaben.	10%

Bei der Konzeption und Erstellung des Portfolios sollten die genannten Bewertungskriterien einschließlich der folgenden Erläuterungen berücksichtigt werden.

Problemabgrenzung/Zielsetzung: Korrekte Reflexion und Implementierung von Data Engineering Konzepten wie Zuverlässigkeit (*reliability*), Skalierbarkeit (*scalability*), Wartbarkeit (*maintainability*), Datensicherheit (*data security*), Data Governance und Datenschutz (*data protection*)

Methodik/Idee/Vorgehen: Berücksichtigung technischer Anforderungen und schlüssige Argumentation für eine konzeptionelle Datenarchitektur

Qualität der Umsetzung: Erfüllung technischer Anforderungen in der Implementierung, Qualität der Umsetzung in Bezug auf Latenzen, Reproduzierbarkeit mit Hilfe von Infrastructure as Code (IaC) auf anderen Systemen, prägnante und umfassende Dokumentation

Kreativität/Richtigkeit: Kreative Lösungsansätze und eigenständige Fehlerbehebung

Formale Anforderungen: Erfüllung formaler Vorgaben zur Abgabe in den jeweiligen Projektphasen (siehe unten).

4. FORMALIA UND VORGABEN ZUR ABGABE

4.1. Bestandteile der Prüfungsleistung

Im Folgenden befindet sich eine Übersicht der Prüfungsleistung Portfolio mit seinen einzelnen Phasen, einzureichenden Einzelleistungen und Feedbackrunden im Überblick. Für die Erarbeitung der Portfolioteile im Rahmen der Prüfungsleistung wird eine Vorlage in PebblePad zur Verfügung gestellt. Die Vorlage ist Bestandteil dieser Prüfungsleistung.

Phase	Zwischenergebnis	Einzureichende Leistung
Konzeptionsphase	Portfolioteil 1	• Konzeptvorstellung in Textform (200 Wörter, ca. 1/2 Seite DIN A4), einzugeben in die Vorlage in PebblePad: Schriftliche Kurzdarstellung der Architektur unter Berücksichtigung der in der Aufgabenstellung beschriebenen Aspekte • Skizze der Datenarchitektur eingefügt als PNG-Datei
		Feedback
Erarbeitungsphase/ Reflexionsphase	Portfolioteil 2	• Erläuterung zur Umsetzung in Textform (ca. 1/2 Seite DIN A4), einzugeben in die Vorlage in PebblePad • Link zum GitHub Repository in einer eingefügten TXT-Datei
		Feedback
Finalisierungsphase	Portfolioteil 3	• Kurzes Abstract (max. 50 Wörter), einzugeben in die Vorlage in PebblePad: Kurze und prägnante Beschreibung, die die Kernpunkte des Lösungsansatzes der implementierten Architektur zusammenfasst • 2-seitiges Full Abstract als PDF: Beschreibung der technischen Aspekte, die die Erfüllung der Vorgaben an die Datenarchitektur sicherstellen, sowie eine persönliche Projektreflexion im Sinne der in der Aufgabenstellung genannten Punkte • Ein Dokument (TXT) mit einem Link zum Git Hub Repository, welches Deinen Code enthält. Dieser Code ist das Endprodukt deines Projektes, d. h. die von dir entwickelte Datenarchitektur ist durch diesen Code reproduzierbar, übertragbar und auf einem anderen Rechner mit Docker-Containern ausführbar. • Ergebnis aus Phase 1 • Ergebnis aus Phase 2
		Feedback + Note

4.2. Formalia zur Abgabe digitaler Dateien

Konzeptionsphase

Empfohlene Hilfsmittel/Software zur Bearbeitung	Git Hub, Docker für eine lokale Maschine, IDE nach Vorlieben (bspw. VS Code)
Zugelassene Dateiformate	Visuelle Skizze der Architektur als PNG
Dateigröße	möglichst gering
Weitere Formalien und Parameter	Dateien sind immer nach Folgendem Muster zu benennen: **Für die prüfungsleistungsrelevanten Abgaben auf PebblePad:** Nachname-Vorname_Matrikelnummer_Kurskürzel _P(hase)Phasennummer Beispiel: Mustermann-Max_12345678_ DLMDWWDE02 _P1

Erarbeitungs-/Reflexionsphase

Empfohlene Hilfsmittel/Software zur Bearbeitung	Git Hub, Docker für eine lokale Maschine, IDE nach Vorlieben (bspw. VS Code)
Zugelassene Dateiformate	Link zu einem Git Hub Repository als TXT
Dateigröße	möglichst gering
Weitere Formalien und Parameter	Dateien sind immer nach Folgendem Muster zu benennen: **Für die prüfungsleistungsrelevanten Abgaben auf PebblePad:** Nachname-Vorname_Matrikelnummer_Kurskürzel _P(hase)Phasennummer Beispiel: Mustermann-Max_12345678_ DLMDWWDE02 _P2

Finalisierungsphase

Empfohlene Hilfsmittel/Software zur Bearbeitung	Git Hub, Docker für eine lokale Maschine, IDE nach Vorlieben (bspw. VS Code)
Zugelassene Dateiformate	Full Abstract als PDF, Link zu einem Git Hub Repository als TXT
Dateigröße	möglichst gering
Weitere Formalien und Parameter	Das Full Abstract ist als druckfähige PDFs in Hi-Res (300dpi Auflösung, CMYK Farbmo-Dus) abzugeben. Achtet bitte bei den Bildern (und ggf. Schriften), die in eurem Dokument verknüpft sind, darauf, dass ihr diese entweder einbettet. Sonst sind eure Dokumente nicht vollständig zu öffnen und damit auch nicht zu beurteilen! Dateien sind immer nach Folgendem Muster zu benennen: **Für die prüfungsleistungsrelevanten Abgaben auf PebblePad:** Nachname-Vorname_Matrikelnummer_Kurskürzel _P(hase)Phasennummer Beispiel: Mustermann-Max_12345678_ DLMDWWDE02 _P3

4.3. Formalia für das Abstract

Umfang	2 Seiten Textteil
Papierformat	DIN A4
Seitenränder	Oben und unten 2cm; links 2cm; rechts 2cm
Schrifttyp	Allgemeiner Text – Arial 11Pkt; Überschriften – 12Pkt, Blocksatz
Zeilenabstand	1,5
Satz	Blocksatz und Silbentrennung
Fußnoten	Arial 10Pkt, Blocksatz
Absätze	Nach gedanklicher Gliederung – 6Pkt Abstand nach Zeilenumbruch
Eidesstattliche Erklärung	Die Abgabe der Eidesstattlichen Erklärung erfolgt in elektronischer Form über myCampus. Davor ist keine Einreichung der Prüfungsleistung möglich. Bitte beachtet hierzu die Anleitung für das Einreichen eines Portfolios in myCampus.

Bei Fragen zur Abgabe des Portfolios wende Dich bitte per Mail an das Prüfungsamt.

Beachte bitte zusätzlich die Nutzungsanleitung zu PebblePad & Atlas!

Viel Erfolg beim Erstellen des Portfolios!