VUKFZBO-731 [Research:reactive@db] Personnel-API um Reactive-ORM erweitern

1) Hibernate-reactive (HR)
   a) How to use
      i) Setting up and configure
         (1) Using HR outside of Quarkus → include hibernate reactive with Vert.x Mysql
         (2) Continue following the documentation
      ii) Mapping relational objects
         (1) Identical to regular Hibernate ORM and other implementations of JPA.
         (2) Custom identifier generators written to work with Hibernate ORM and JDBC will not work in the reactive environment
         (3) Any JPA AttributeConverter works in Hibernate Reactive.
      iii) Perform persistance operation for transactions
         (1) manage transactions and reactive sessions
            (a) the same session should be reused across multiple persistence operations within a single reactive stream representing a certain transaction or unit of work, but a session shouldn't be shared between different concurrent reactive streams.
            (b) The Session interface has methods with the same names as methods of the JPA EntityManager
            (c) Difference in HR: in the reactive API, each of these methods returns its result in a non-blocking fashion via a Java CompletionStage (or Mutiny Uni).
            (d) For JPA criteria queries, first obtain the CriteriaBuilder using SessionFactory.getCriteriaBuilder(), and execute query using Session.createQuery()
         (2) construct reactive streams by chaining persistence operations invoked on the reactive session. Choose between these two APIs

Purpose	**Java** `CompletionStage`	**Mutiny** `Uni`
Chain non-blocking operations	`thenCompose()`	`chain()`
Transform streamed items	`thenApply()`	`map()` and `replaceWith()`
Perform an action using streamed items	`thenAccept()`	`invoke()` and `call()`
Perform cleanup (similar to `finally`)	`whenComplete()`	`eventually()`

(3) fetch and prepare data needed by the UI

(4) handle failures

b) Advantages

i) Easy to get started (Docs claim)

ii) This programming paradigm holds the potential for improved scalability, and more controlled degradation under peak load, in some runtime scenarios. (Docs claim)

iii) Identical object mapping to regular hibernate ORM and other JPA

iv) A lot of steps are similar to plain JPA

v) Active community and detailed documentation

vi) Existing hibernate mapping can be used (in personnel unapplicable)

c) Disadvantages

i) An immediate performance gain in performance tests might not be instantly noticeable. Many programs will not benefit from the programming model, and those which do benefit might only benefit in very specific load scenarios.

ii) Dependant on Vert.x, extra learning for developer is required.

iii) Configuration might take some time (doesn't seem simple)

d) Notes

i) It seems possible to keep the existing R2DBC implementation and using HR at the same time with separate configuration. Might need to test which versions are compatible.

2) Ebean ORM

a) ORM, not reactive (Reactive support only via R2DBC)

3) Micronaut Data Reactive

a) ORM, reactive, only for micronaut application

4) Vert.x MySQL

a) Reactive, not ORM

5) JOOQ

a) Not reactive (reactive support only via R2DBC)

b) jOOQ is an ORM-alternative that is relational model centric rather than domain model centric like most ORMs.