Bonifico

Introduzione

Una generica operazione di bonifico prevede che il cliente inserisca dei dati obbligatori e altri facoltativi:

- alcuni dati sono ad inserimento libero, come ad esempio nome e cognome del destinatario;
- alcuni con validazione effettuata lato frontend, come ad esempio l'importo numerico o la data;
- altri lato backend, ad esempio il saldo del cliente e la validità dell'IBAN.

Per gli esercizi, vengono proposti dei json di esempio relativi a 3 API:

- **bonifico.prepare** : questa API si occupa di dare i dati che possono servire al frontend per agevolare la selezione da parte dell'utente e/o per presentare dei dati;
- **bonifico.verify** : questa API si occupa di fare le verifiche sul saldo;
- **bonifico.execute** : questa API si occupa di effettuare l'operazione.

API	Http method	Path
bonifico.prepare	GET	`/private/cliente/{idCliente}/conto/{idConto}/bonifico/prepare`
bonifico.verify	POST	`/private/cliente/{idCliente}/conto/{idConto}/bonifico/verify`
bonifico.execute	VERIFY	`/private/cliente/{idCliente}/conto/{idConto}/bonifico/{idTransazione}/execute`

Scenario 1

Il cliente ha 130000 euro sul conto e prova a fare un bonifico di 12 euro, l'operazione va a buon fine.

Sequenza di esempio:

bonifico.prepare

- viene chiamata l'API *bonifico.prepare* senza parametri se non nel path `http://localhost:8080/private/cliente/101132/conto/0001234566/bonifico/prepare`;
- la response (file *sc1.bonifico.prepare.RESPONSE.json)* indica la data odierna e la data massima per l'ordine di bonifico;

bonifico.verify

- viene chiamata l'API *bonifico.verify* con un body (file *sc1.bonifico.verify.REQUEST.json*) contenente importo, beneficiario, iban e causale del bonifico; tale API si occupa di verificare che l'importo sia minore del disponibile e, in caso positivo, memorizza i dati della transazione e restituisce un codice identificativo della stessa `http://localhost:8080/private/cliente/101132/conto/0001234566/bonifico/verify`;
- la response (file *sc1.bonifico.verify.RESPONSE.json*) ritorna l'importo, le commissioni e l'identificativo della transazione;

bonifico.execute

- viene chiamata l'API *bonifico.execute* senza parametri se non nel path `http://localhost:8080/private/cliente/101132/conto/0001234566/bonifico/c5456ec0-96cb-4ab0-abf8-9e2ff9b54ed0/execute`;
- la response (file *sc1.bonifico.execute.RESPONSE.json*) contiene dei messaggi legati al successo dell'operazione e un codice relativo al bonifico.

Scenario 2

Il cliente ha 130.000 euro sul conto e prova a fare un bonifico ad 100.000 euro, ma viene bloccato in quanto il limite massimo è attualmente configurato a 50.000 euro (controllo lato backend).

Sequenza di esempio:

bonifico.prepare

- viene chiamata l'API *bonifico.prepare* senza parametri se non nel path `http://localhost:8080/private/cliente/101132/conto/0001234566/bonifico/prepare`;
- la response (file *sc2.bonifico.prepare.RESPONSE.json)* indica la data odierna e la data massima per l'ordine di bonifico;

bonifico.verify

- viene chiamata l'API *bonifico.verify* con un body (file *sc2.bonifico.verify.REQUEST.json*) contenente importo, beneficiario, iban e causale del bonifico; tale API si occupa di verificare che l'importo sia minore del disponibile e, in caso negativo, restituisce un messaggio parlante `http://localhost:8080/private/cliente/101132/conto/0001234566/bonifico/verify`;
- la response (file *sc2.bonifico.verify.RESPONSE.json*) ritorna il messaggio di warning e non il codice della transazione, impedendo di fatto la prosecuzione dell'operazione.

Output atteso

L'output degli esercizi verrà valutato indipendentemente dalla completezza di quanto svolto: è obbligatorio un documento che descriva l'approccio e le analisi del progetto; sono ben accetti eventuali rimandi a tecnologie/articoli/esempi presi come riferimento.
Non si pongono veti all'utilizzo di framework.
Si richiede che il sorgente software sia caricato su github o equivalenti.

Dal punto di vista del backend

1. sviluppo metodo per gestire input/output di *bonifico.prepare*, che deve:
   - restituire la data odierna;
   - restituire una data limite inserimento che corrisponde alla odierna prima più 30 giorni;
2. sviluppo metodo per gestire input/output di *bonifico.verify*, che deve:
   - recuperare il saldo del cliente, basandosi sul codice cliente e sul codice conto;
   - verificare che l'importo del bonifico sia inferiore a quanto disponibile;
   - se la validazione è corretta, bisogna generare una chiave e inserire su file i dati del cliente che sta facendo il bonifico (codice cliente, codice conto), oltre a importo e data per il bonifico;
3. sviluppo metodo per gestire input/output di *bonifico.execute*, che deve:
   - recuperare da file quanto generato dalla bonifico.verify;
   - aggiornare il saldo del cliente;
   - generare un numero relativo all'operazione eseguita e conservare le informazioni relative all'operazione.

NOTA: Per ciò che riguarda il recupero/aggiornamento di dati, si possono utilizzare indifferentemente dei file o un db.

Dal punto di vista del frontend

Indipendentemente dalla tecnologia utilizzata, prevedere uno strato di presentazione in cui è possibile:

1. realizzare una form, eventualmente valorizzata con i campi restituiti dalla *bonifico.prepare*;
2. inserire i dati utili alla *bonifico.verify*, effettuando:
   - validazione dei campi obbligatori nome, cognome, importo, iban e data
   - validazione sintattica del campo data e importo
   - bloccando l'invio della form se una validazione non è superata
3. gestire l'output della *bonifico.verify*:
   - visualizzando una pagina di riepilogo da cui chiamare l'API di *bonifico.execute* o (in base al test)
   - visualizzando il messaggio di warning/errore scaturito dalla bonifico.verify;
4. gestire l'output della *bonifico.execute*.

NOTA: Per ciò che riguarda il recupero/aggiornamento di dati, si possono utilizzare dei mock e/o dei dati cablati.