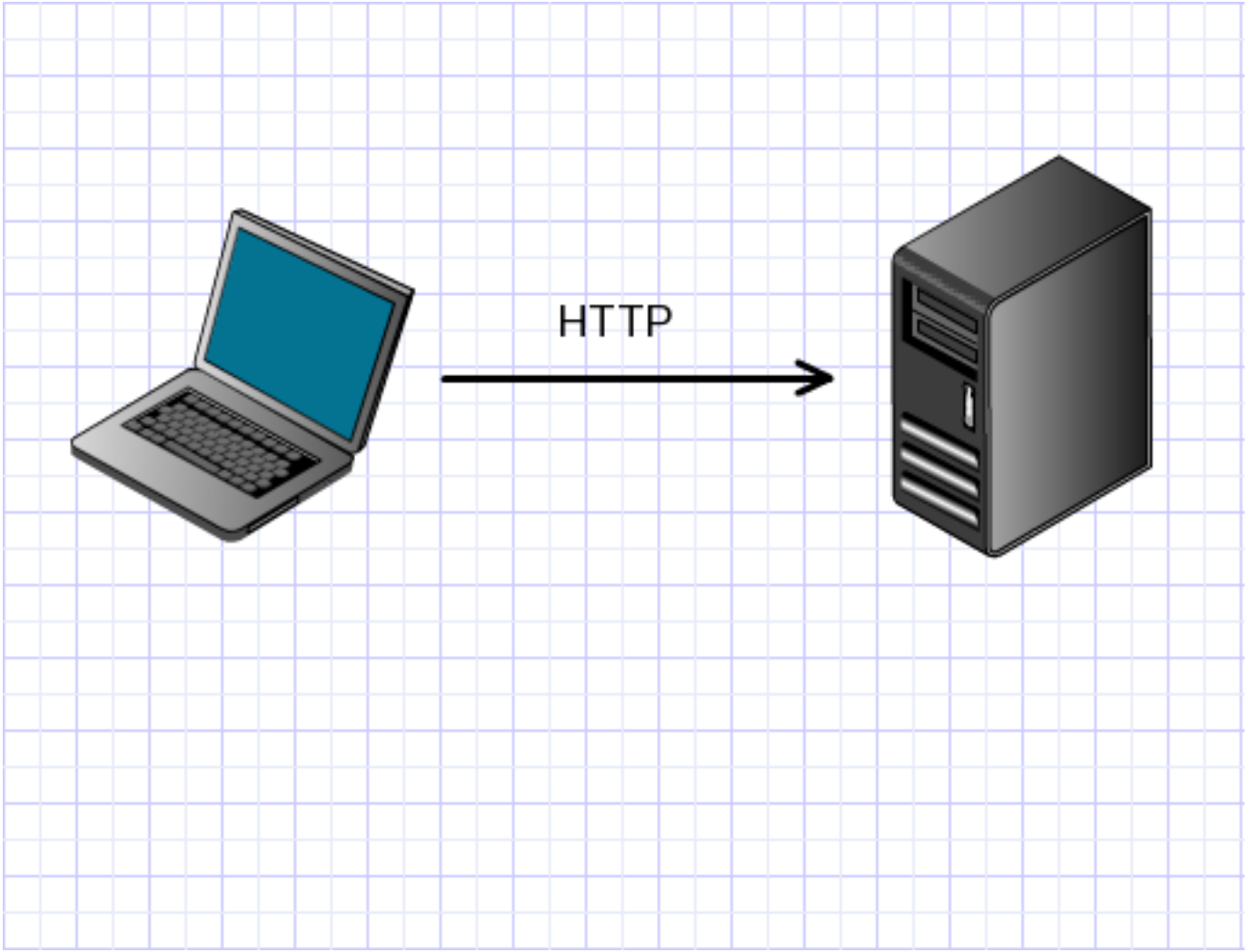
Messaging is not just for investment banks! (+ web is not the only way)

Gojko Adzic
http://gojko.net
gojko@gojko.com
twitter.com/gojkoadzic

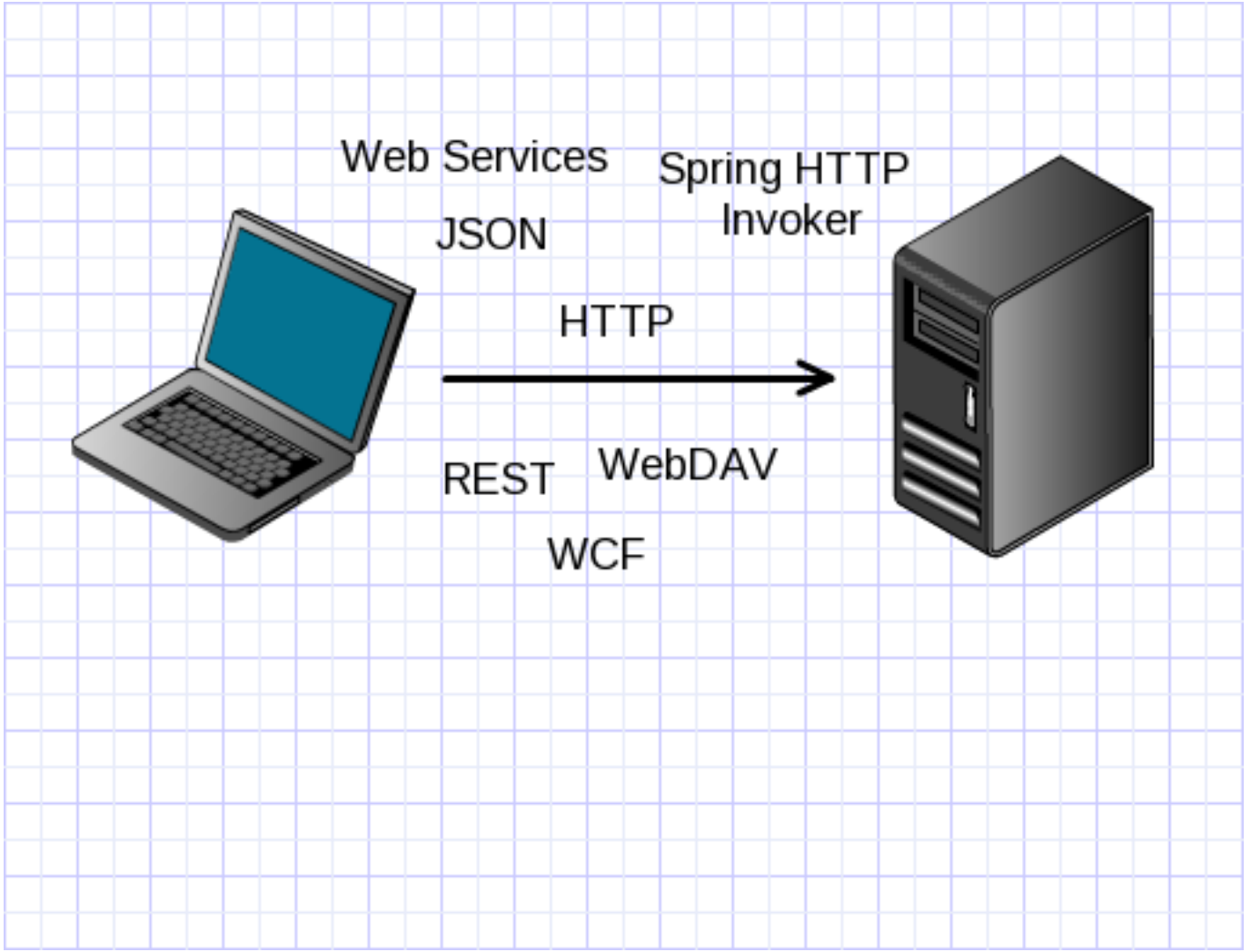
Web is now ubiquitous

... we use it to display content, to share data, for remote procedure calls, to integrate systems...

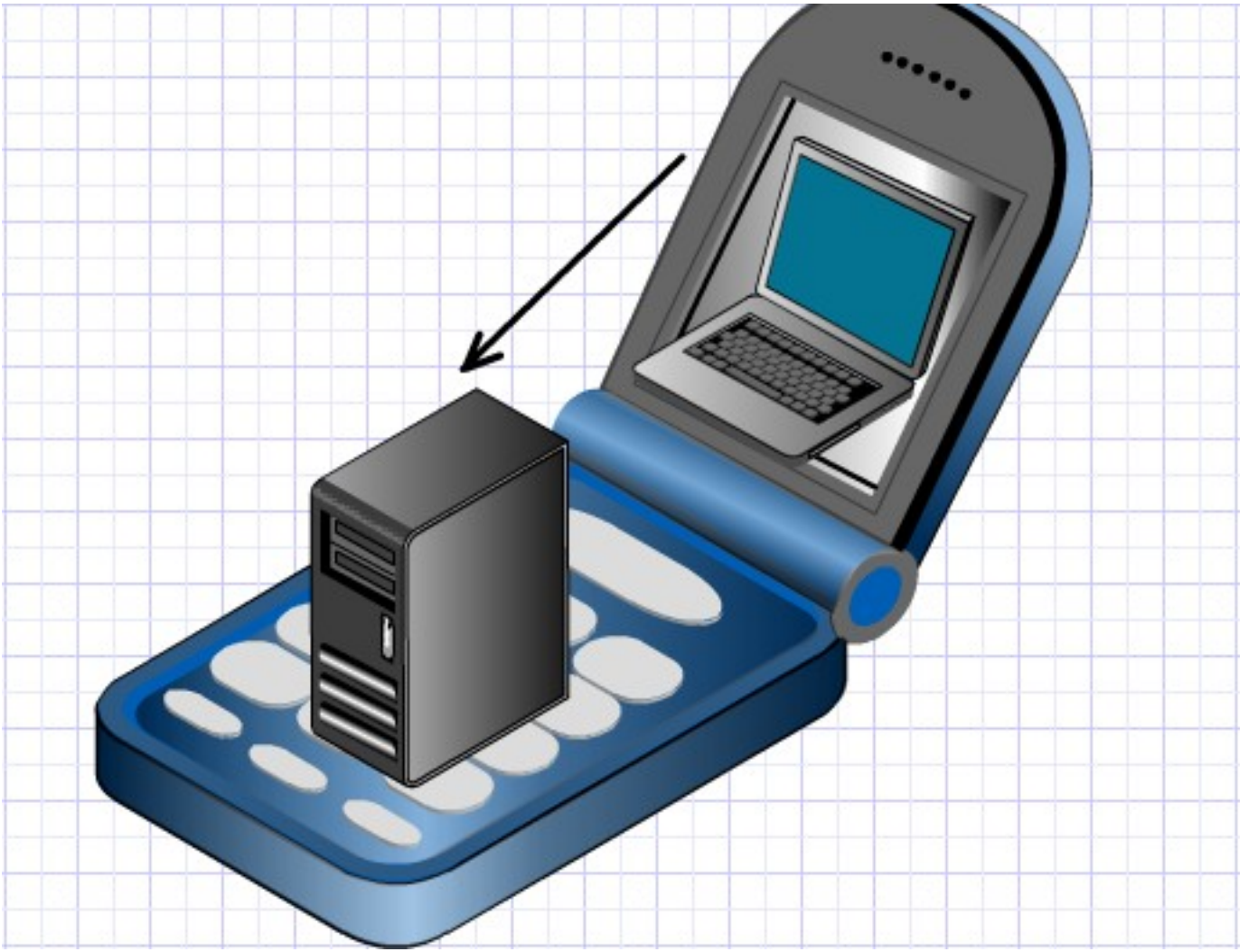
It started like this...



And simply exploded...



But why this???



HTTP distribution/integration

- Easy to use
- Cross-platform
- Clustering and load balancing
- Almost never blocked by firewalls
- Even then works with proxying
- Stateless
- Synchronous
- Unreliable/Non transactional
- High latency

Web technologies are not for everything...

... sometimes other stuff can save a lot of effort

http://www.flickr.com/photos/33453508@N02/

HTTP distribution/integration

- Easy to use
- Cross-platform
- Clustering and load balancing
- Almost never a security problem
- Even then works with proxying
- **Stateless**
- **Synchronous**
- **Unreliable/Non transactional**
- **High Latency**

Messaging

- Application integration pattern
- Data transformation, routing, resilience, high performance, high throughput
- Message oriented middleware(MOM)

http://www.flickr.com/photos/wirenine/

http://www.flickr.com/photos/17675967@N02/


LANERS
BUS
LINES

http://www.flickr.com/photos/stewf/

Messaging

- Applicatio integra n pattern
- Data transfo , routing, resilience, high performance throughput
- Message o ted m dleware(MOM)
- Event driven processing
- Split workflows into several asynchronous parts
- Share data instead of functionallity
  - **But use data to invoke actions!**

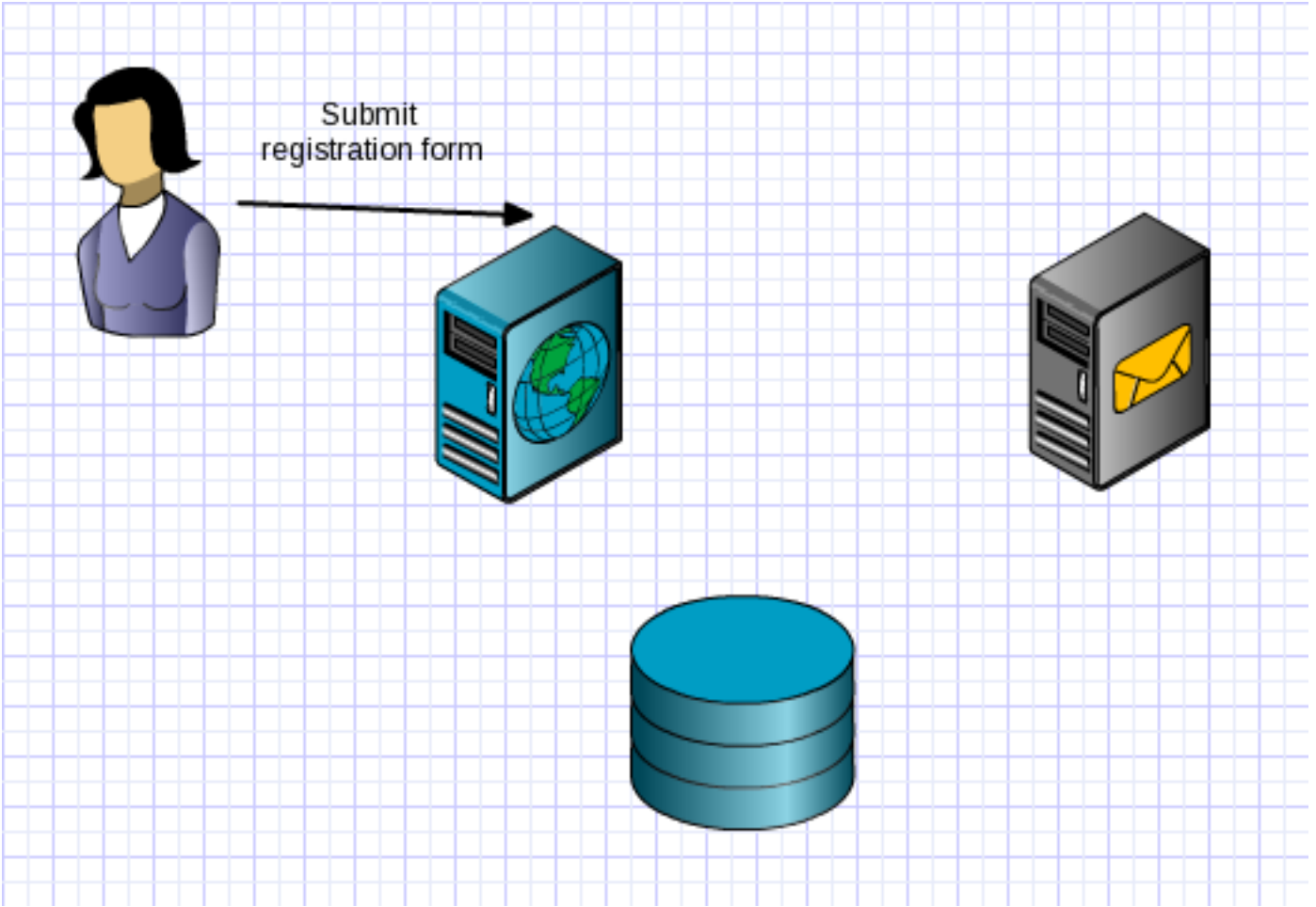
Not just for multi-billion enterprises

- Some really lightweight solutions out there
- Can even run everything on a single machine
- Typical web sites can benefit from this approach as well

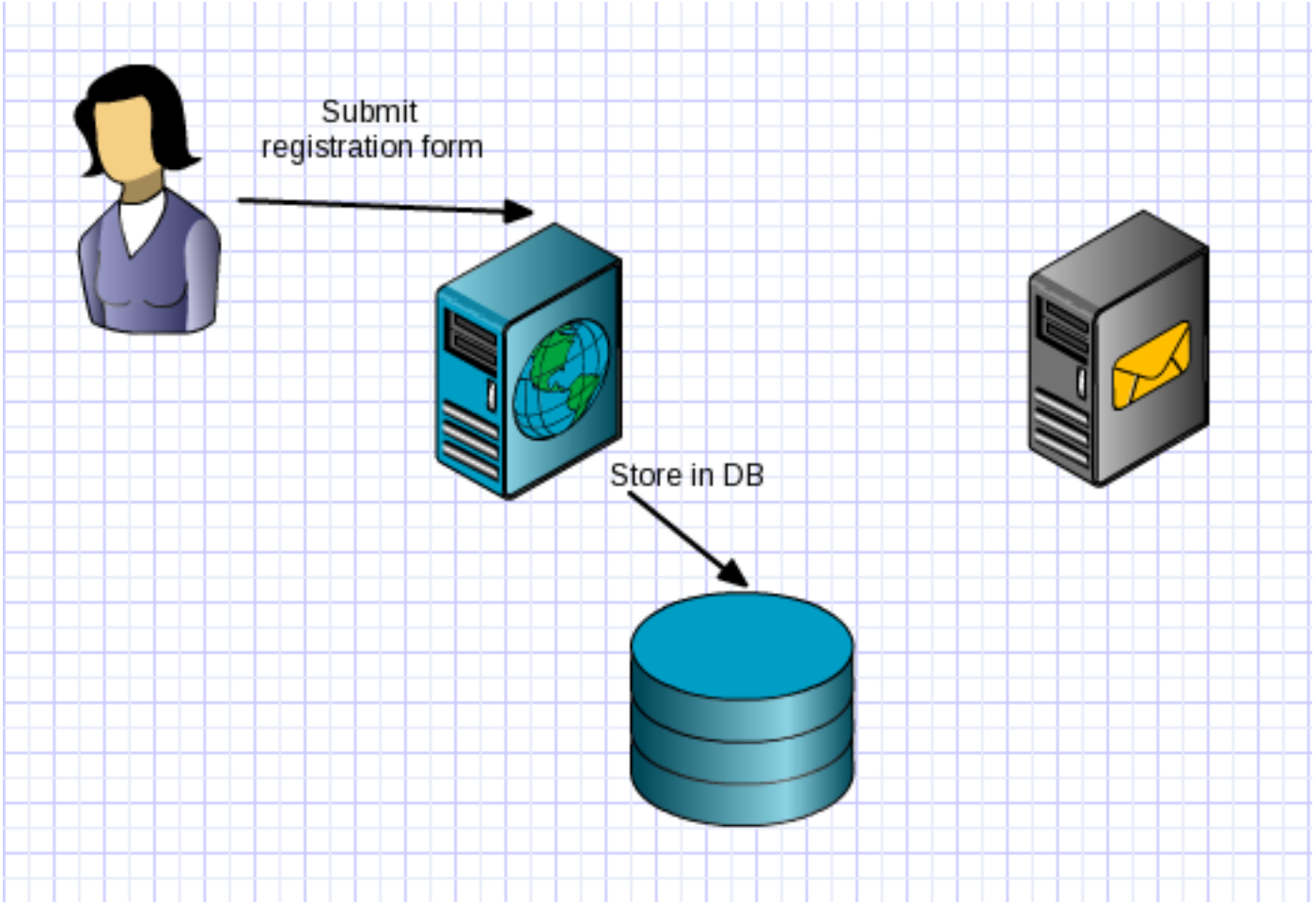
Benefits

- Better isolation
  - Easier scaling
  - Better performance
- Resilience
- Responsiveness
- Better resource usage

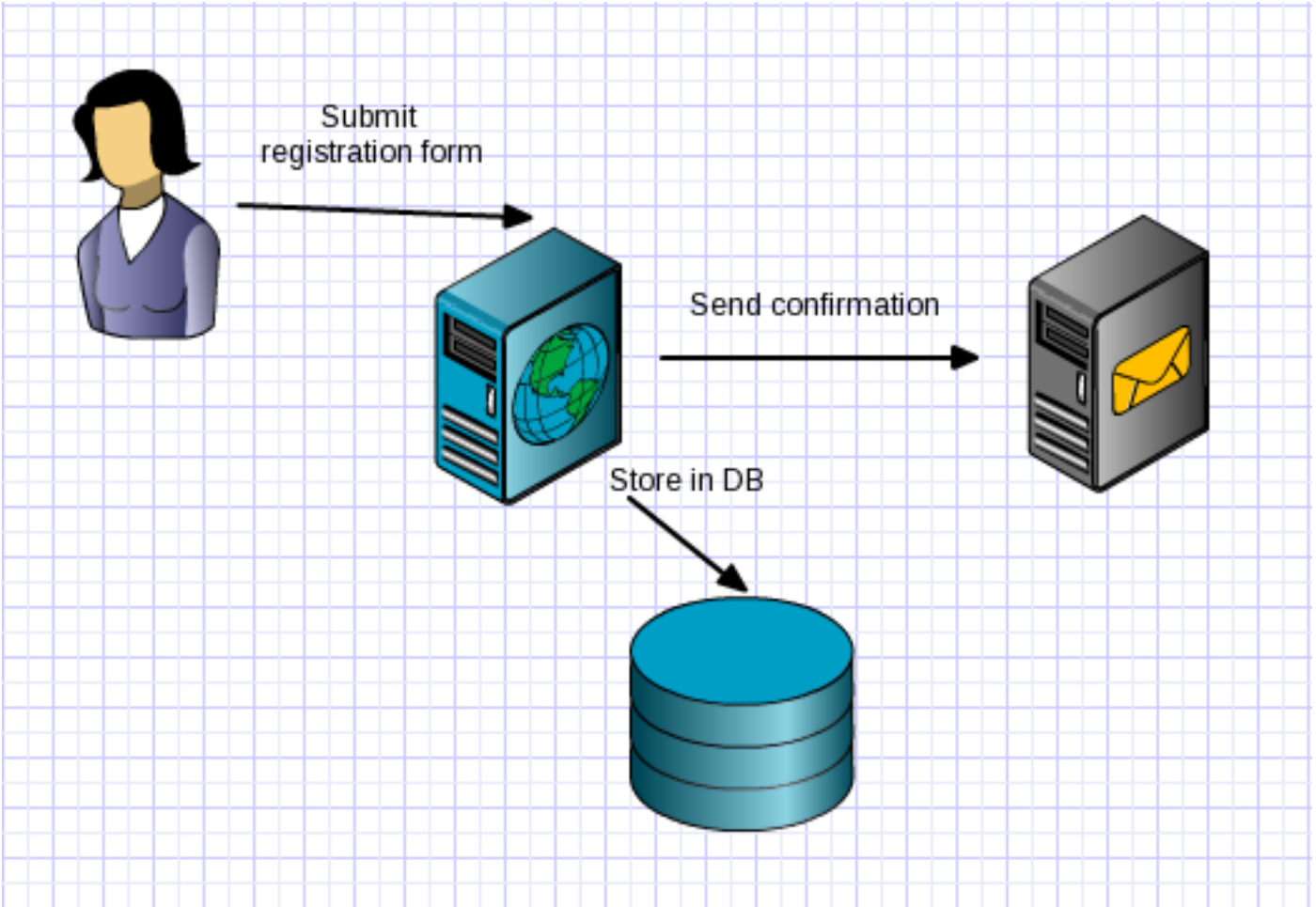
Case Study #1: E-mail after registration/order



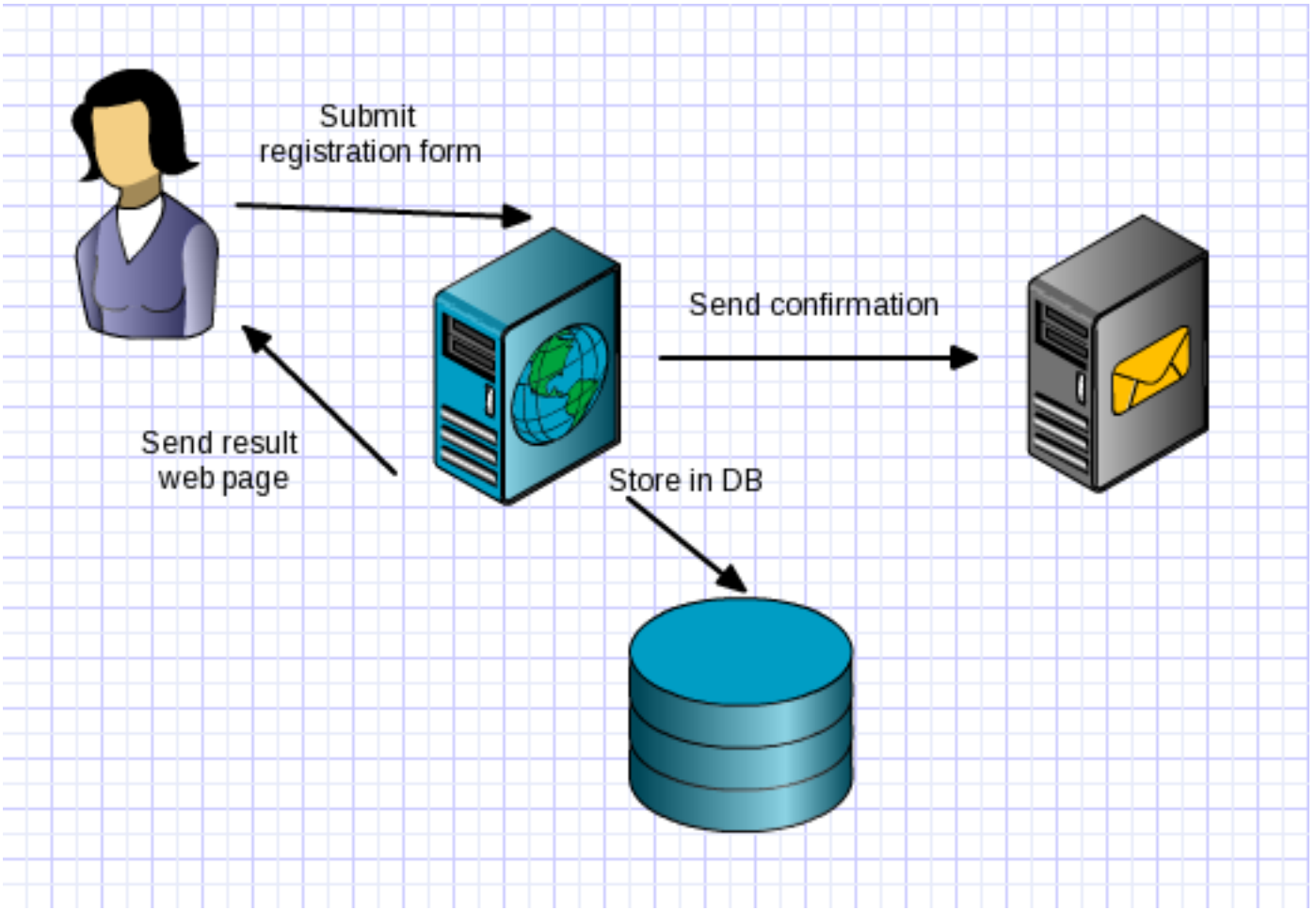
Case Study #1: E-mail after registration/order



Case Study #1: E-mail after registration/order



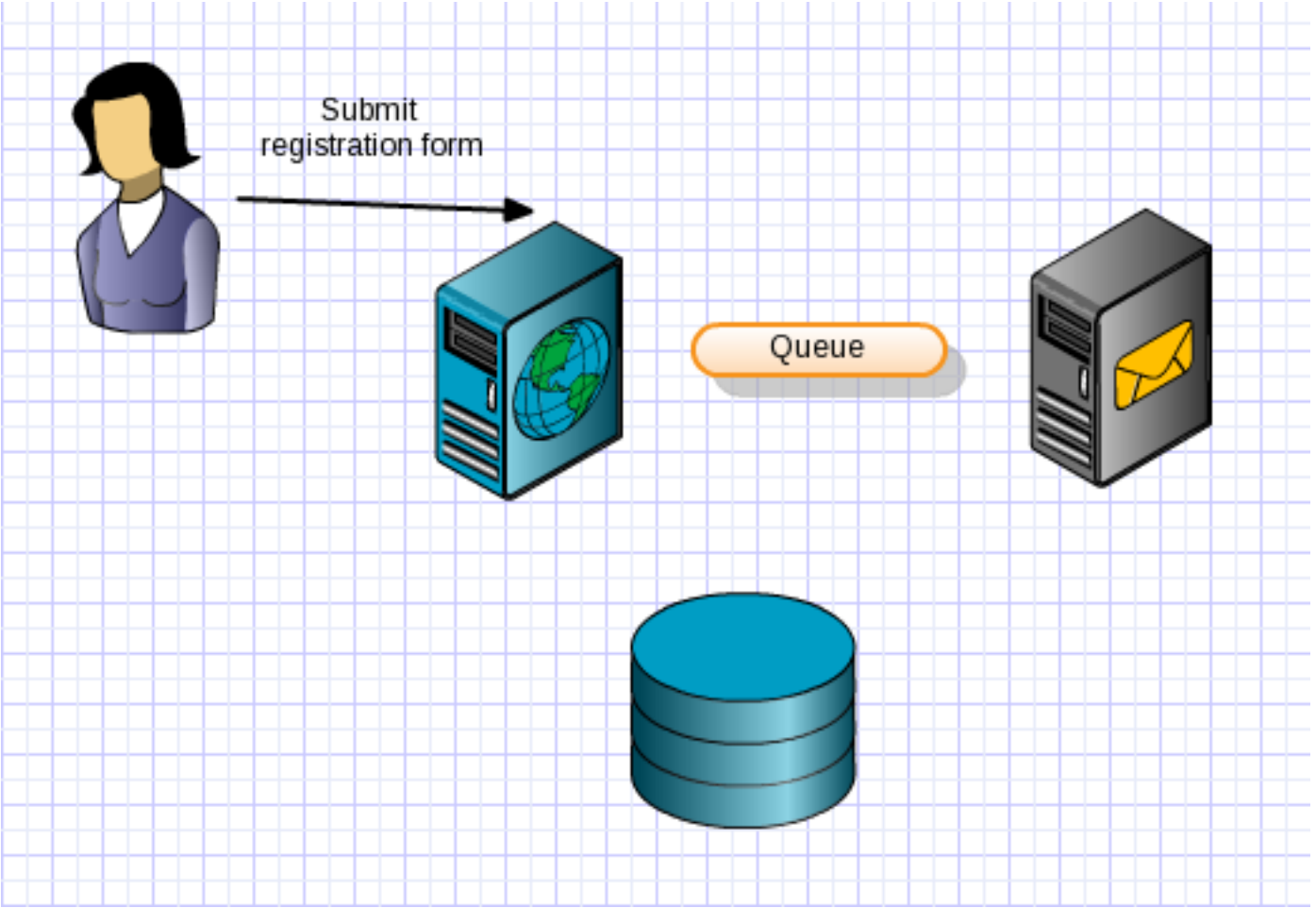
Case Study #1: E-mail after registration/order



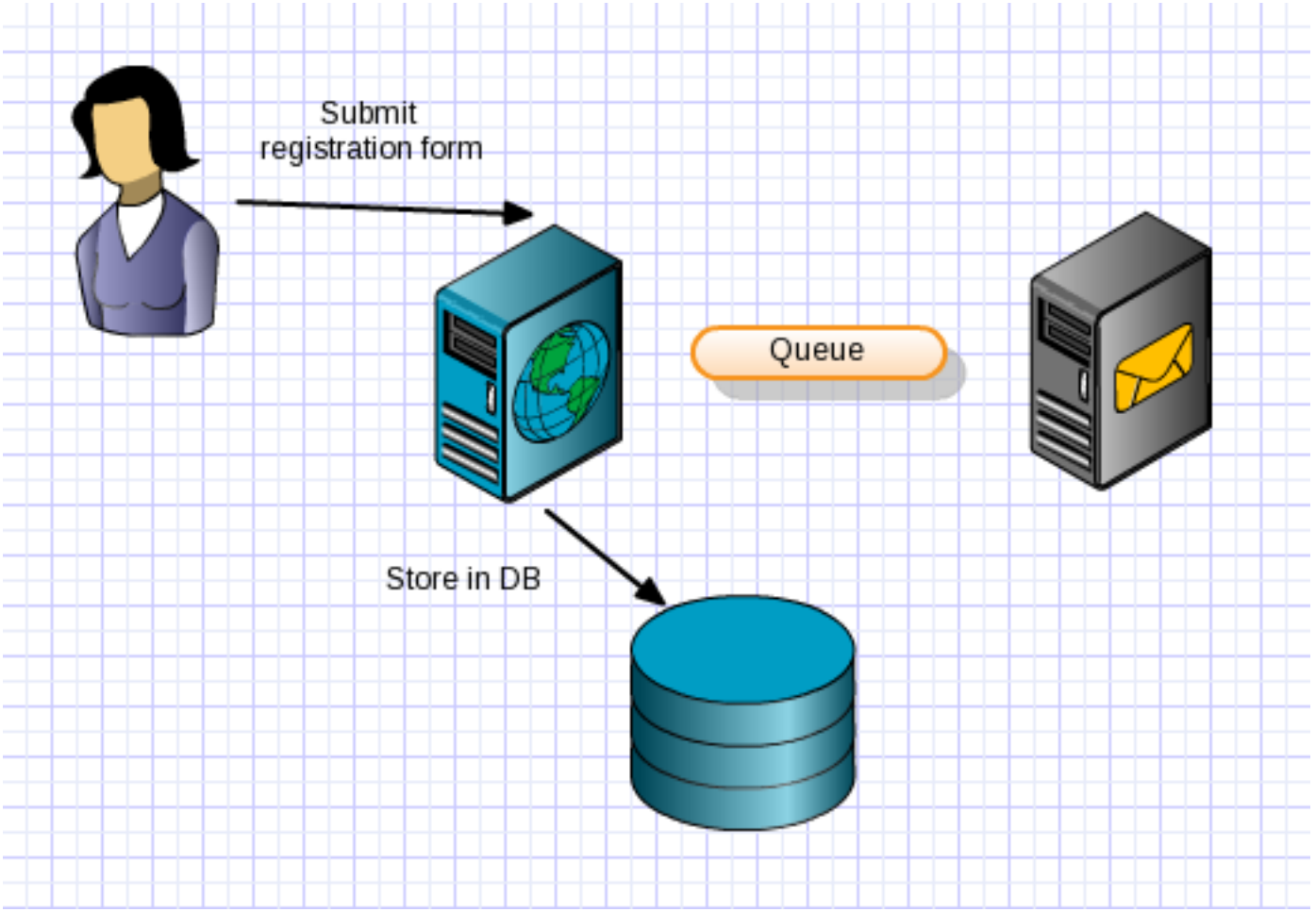
Problems

- DNS/Networking issues
- SMTP rules (external verification, spam filters)
- How do we test this?
- How do we guarantee that the message gets through?
  - What if it doesn't
- What if the DB rolls back?

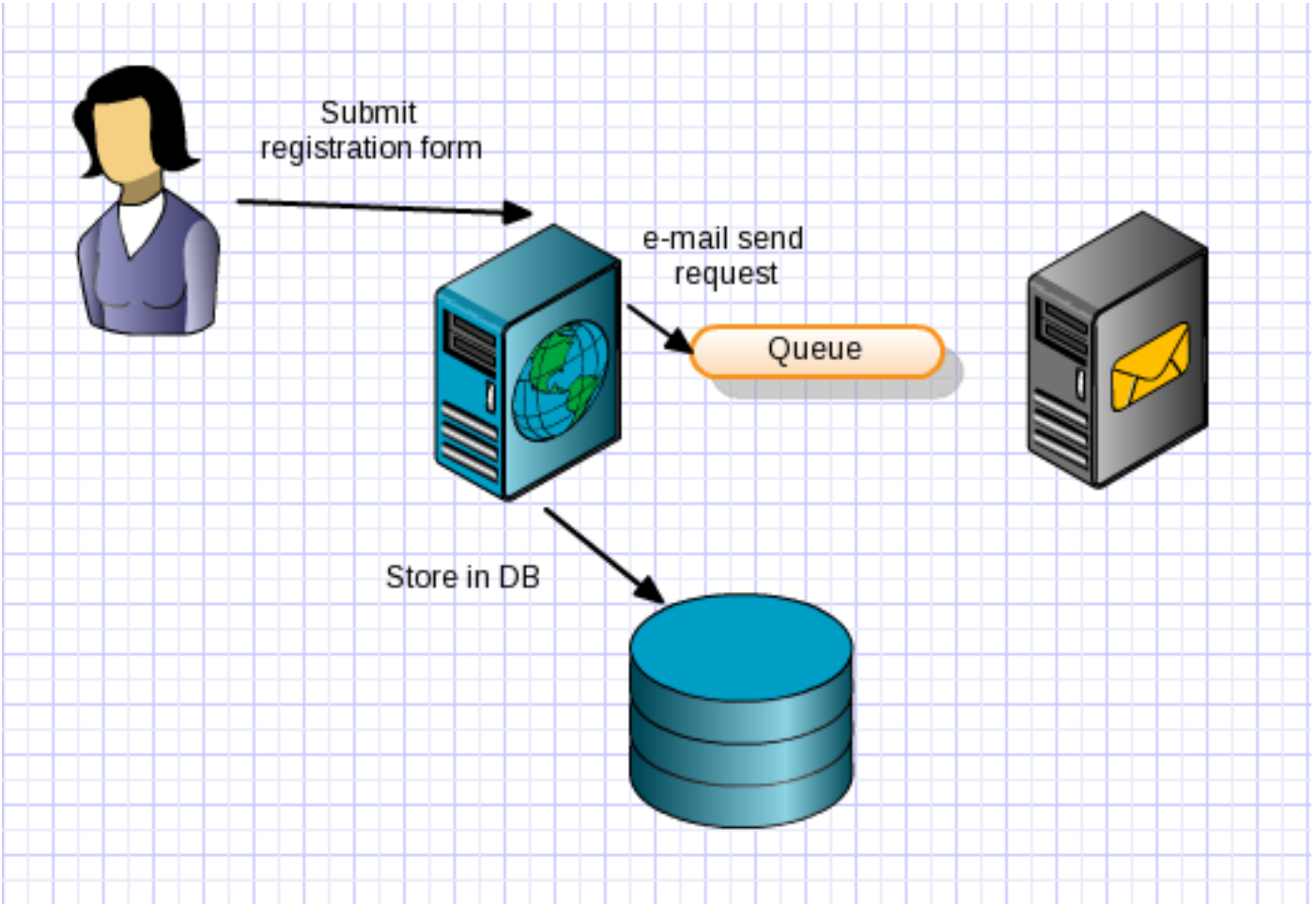
An alternative approach



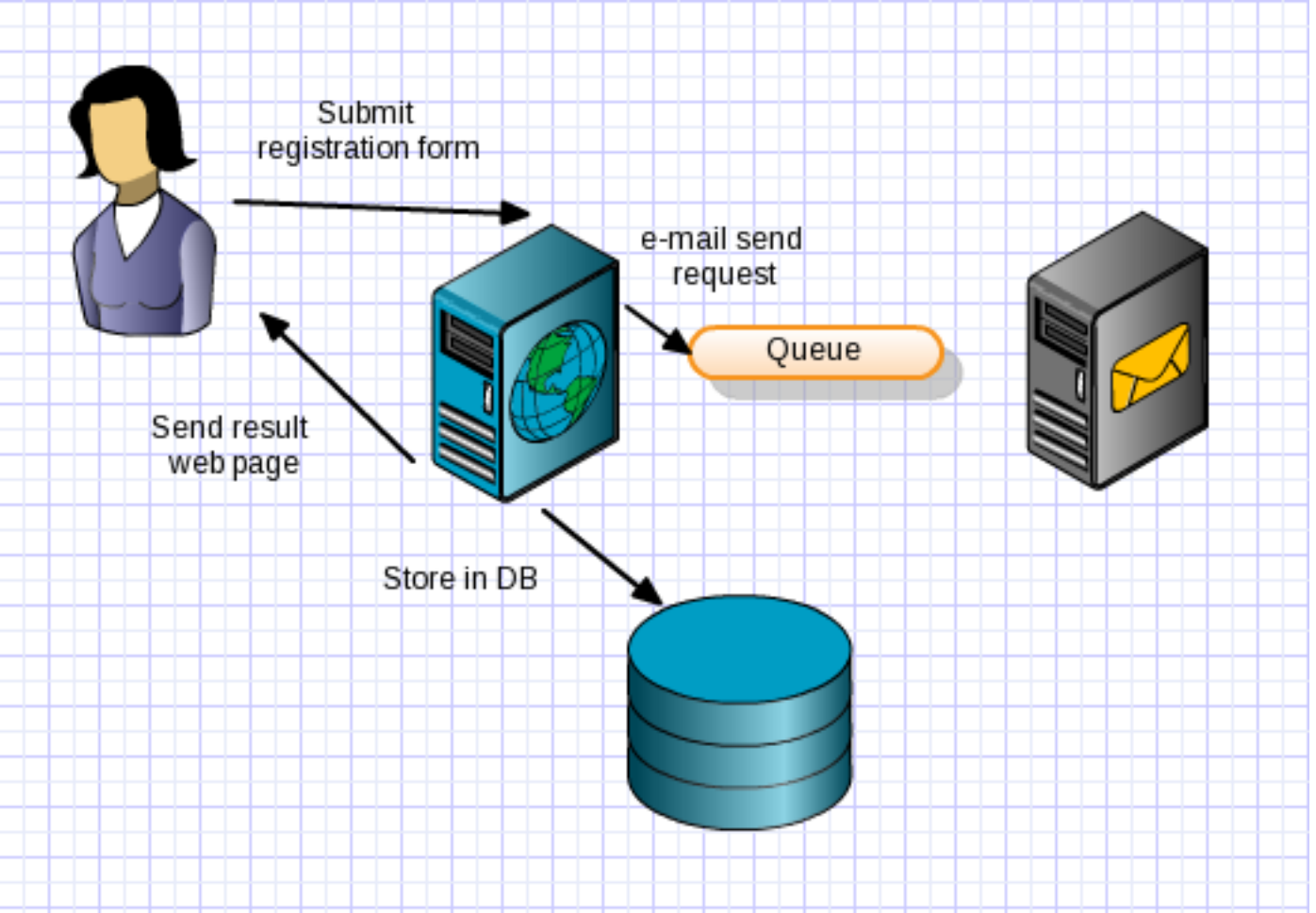
An alternative approach



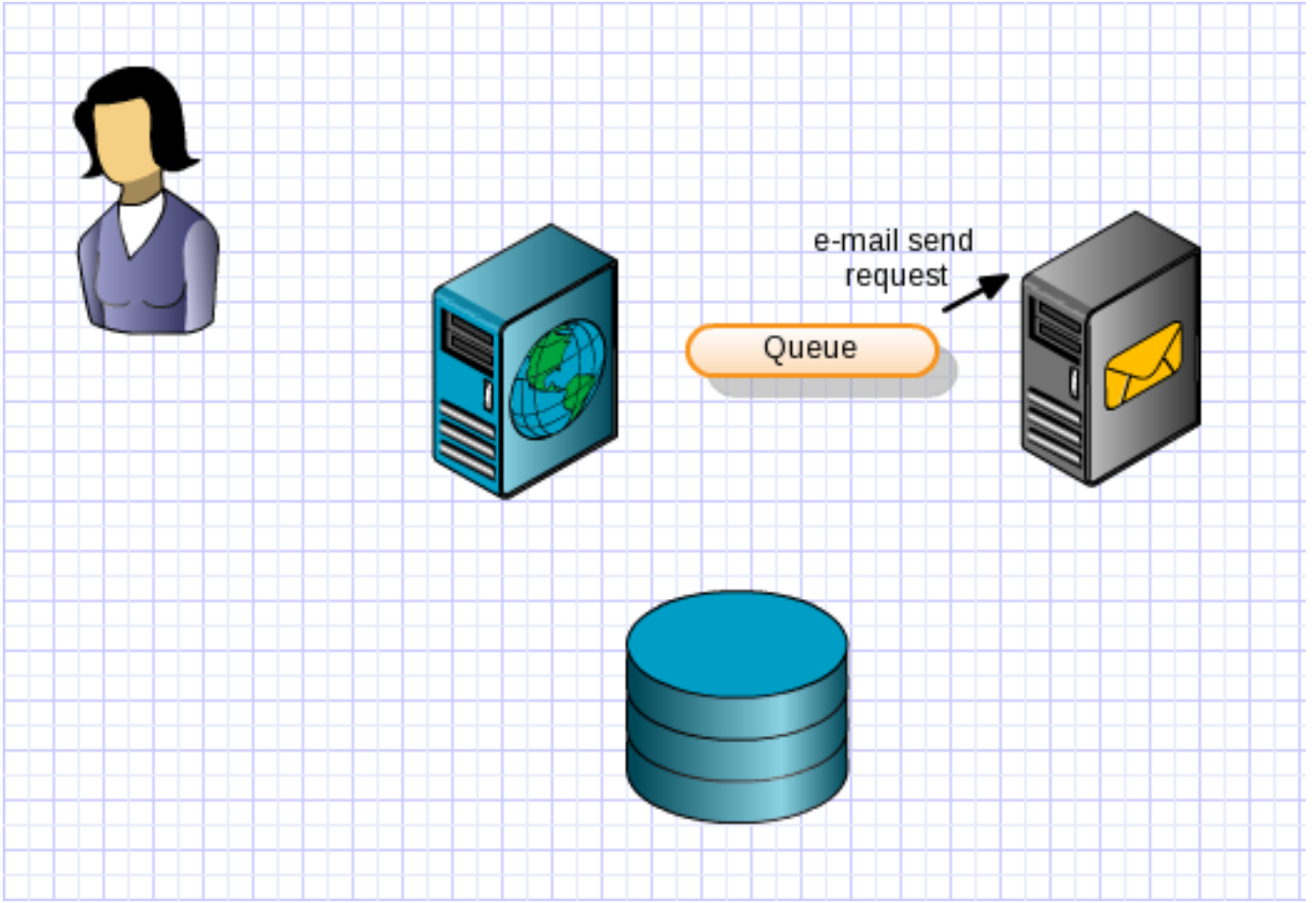
An alternative approach



An alternative approach



An alternative approach



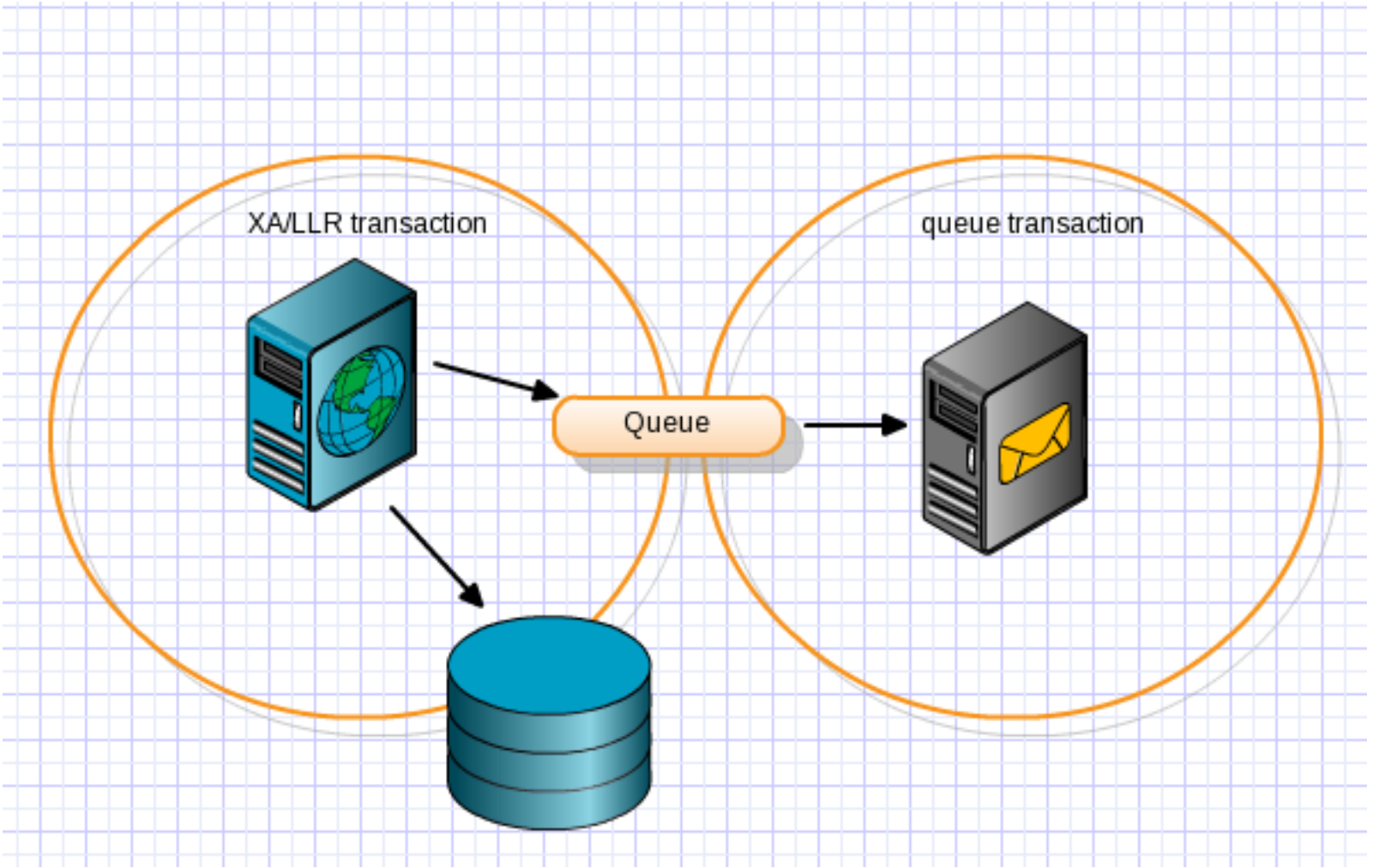
Why is this better?

- External system problems don't affect user registration
- SMTP rules don't slow it down
- If e-mail sending fails, we can easily re-send later

Key Difference:

The first part of the process succeeds without waiting for the second
- but the second is guaranteed to happen

Transactional guarantee



How do we test this?

- Mock queue/In memory implementation
- Process registration and look at the queue contents
- Easily unit testable
- **Focuses the test on what is really important**

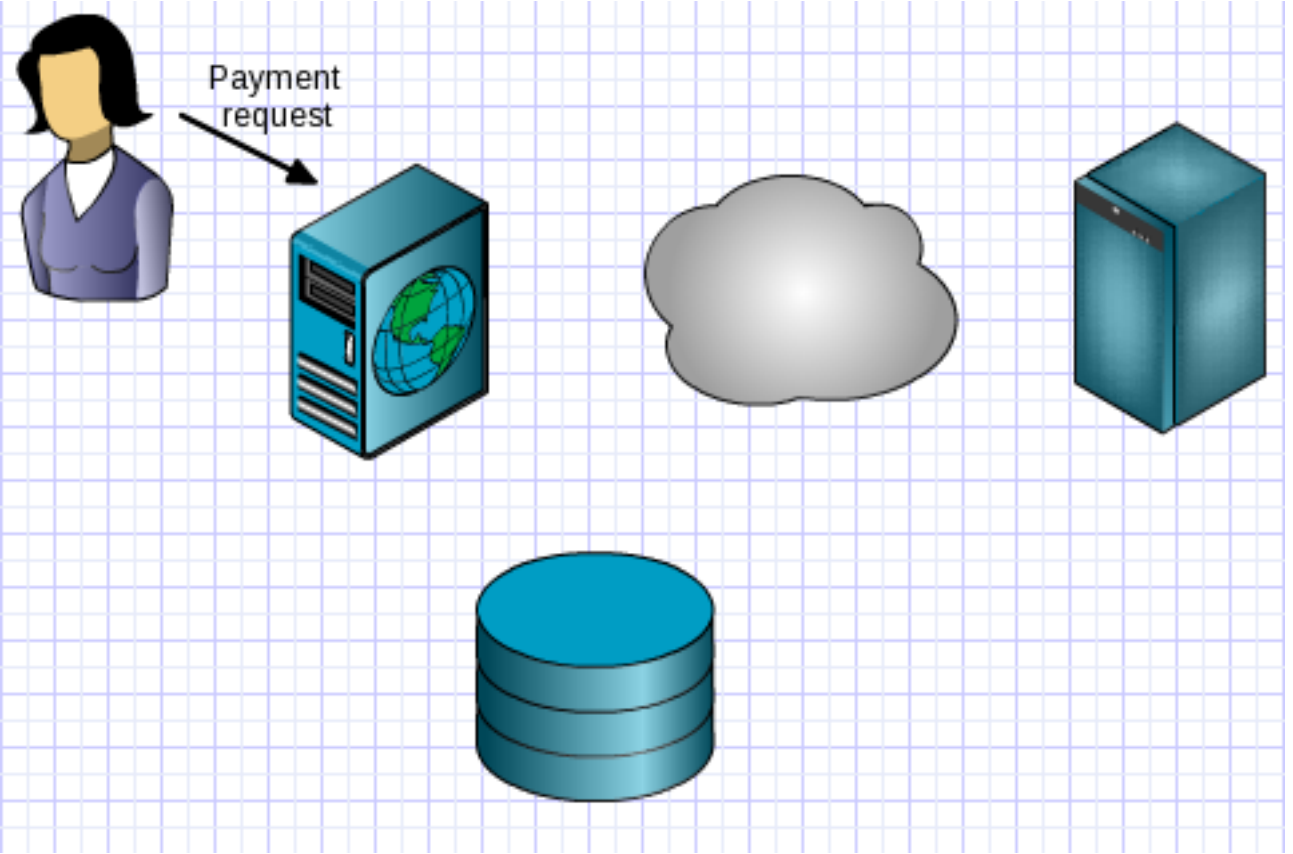
Publish/Subscribe (Fire & Forget)

http://www.sxc.hu/photo/1084274

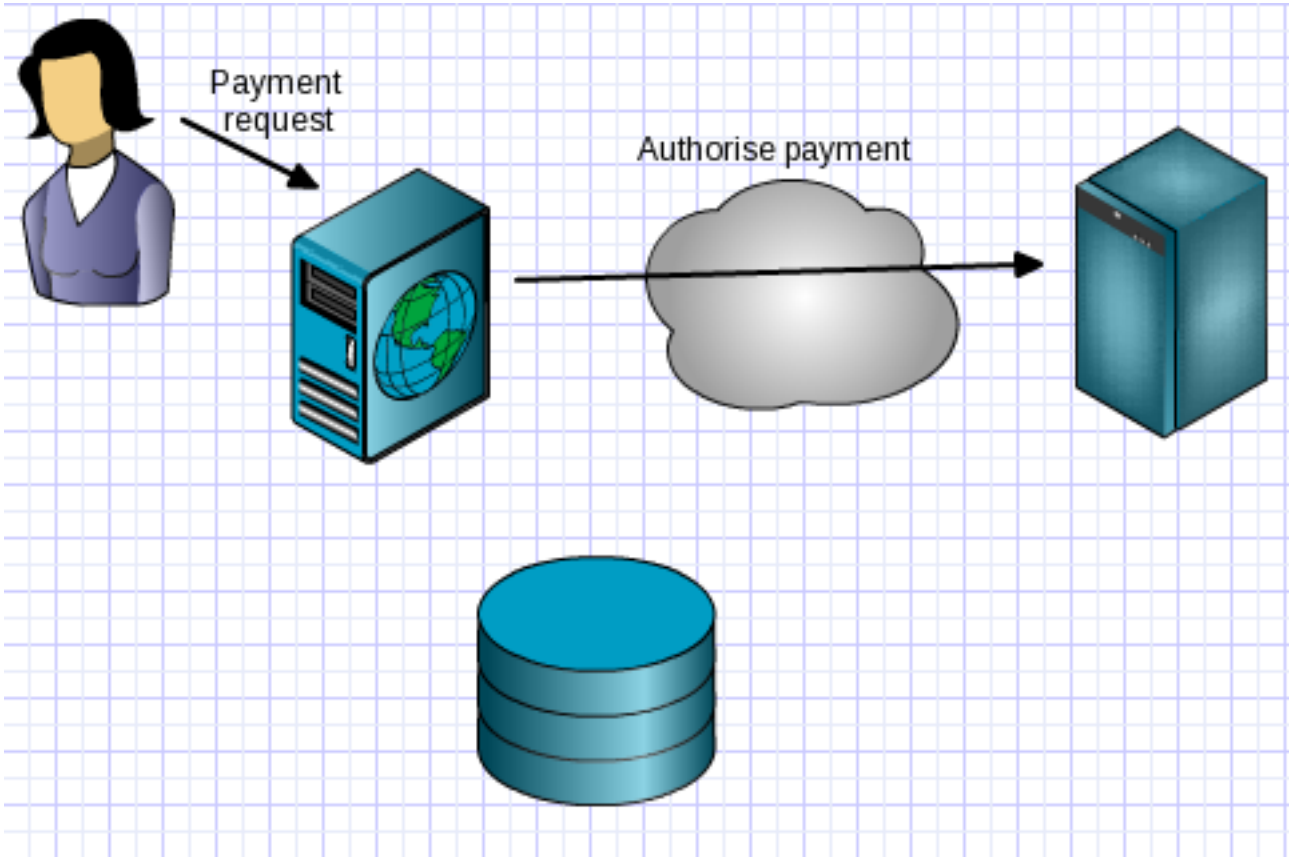
Publish/Subscribe (Fire & Forget)

- When you need to talk to external systems but don't really need to wait for them to finish
- Batch/overnight processing
- Decouple processes so that they can be performed asynchronously
- Effectively observer over messaging
- Option to have multiple subscribers (observers)

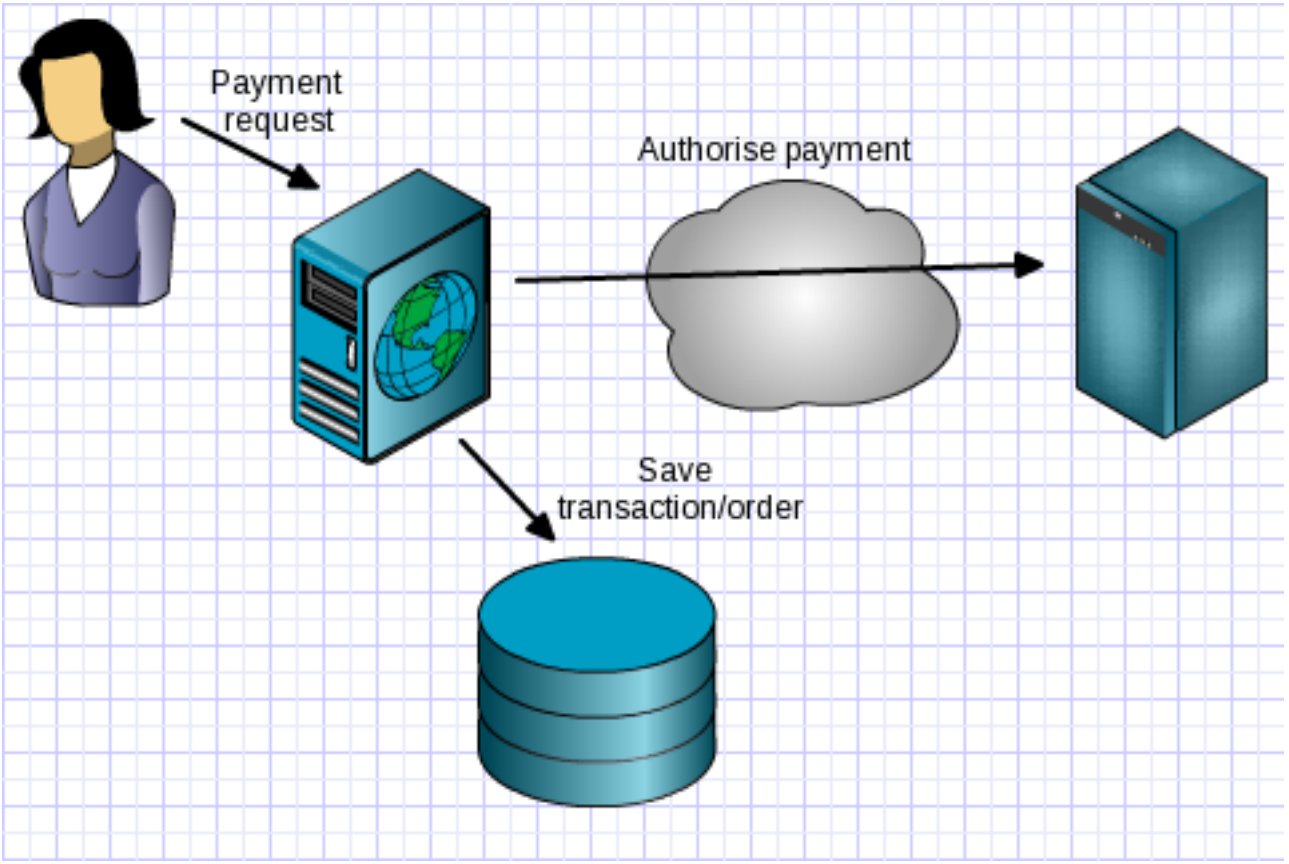
Case study #2: Credit card processing



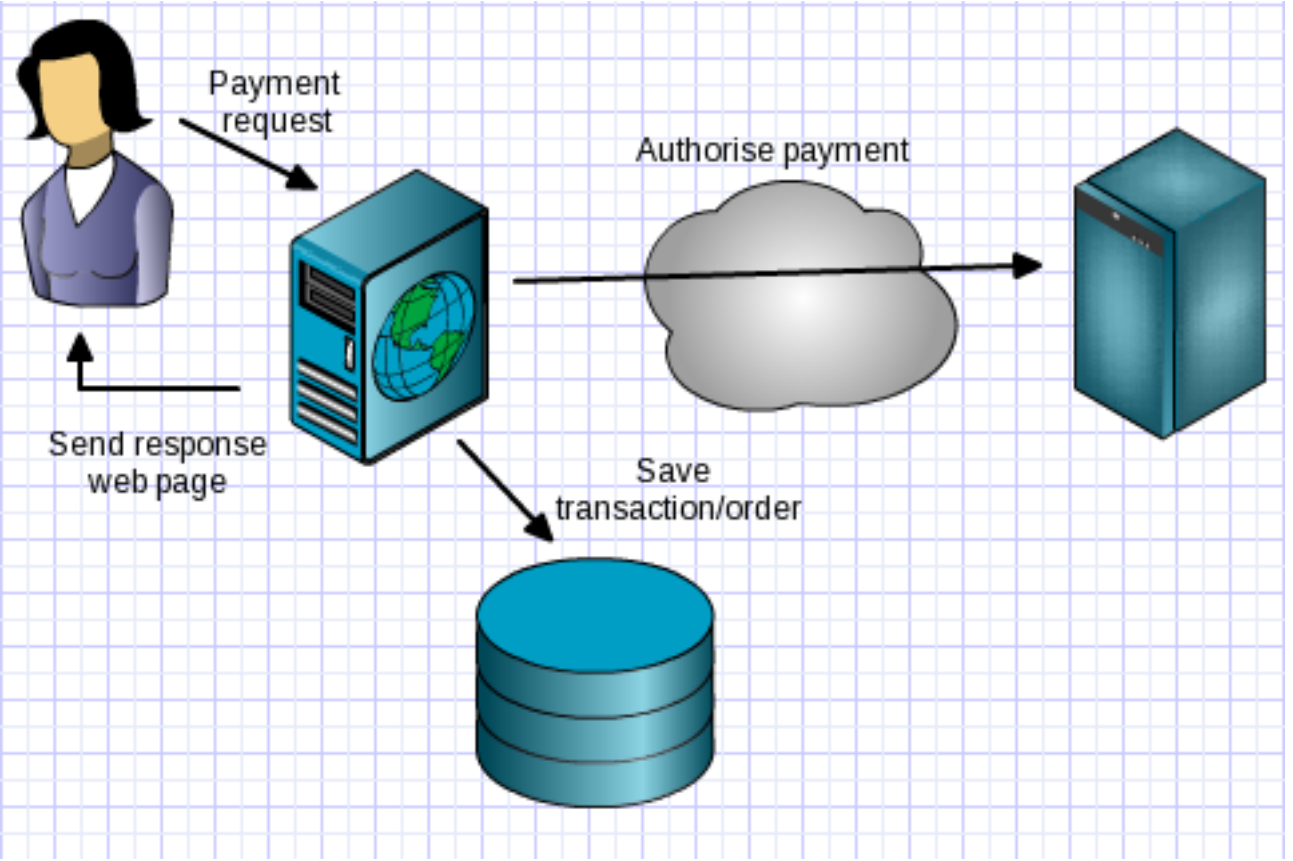
Case study #2: Credit card processing



Case study #2: Credit card processing



Case study #2: Credit card processing

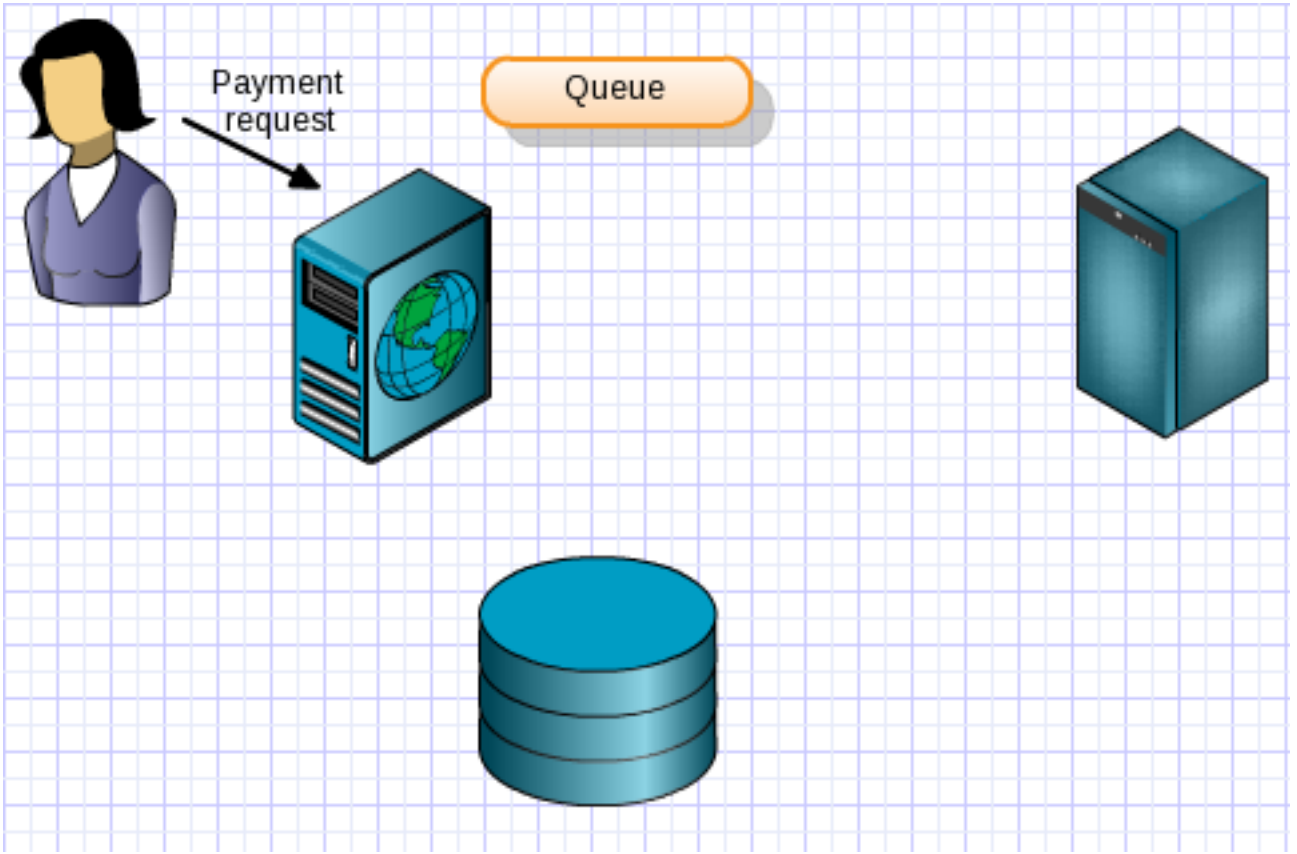


What could possibly go wrong?

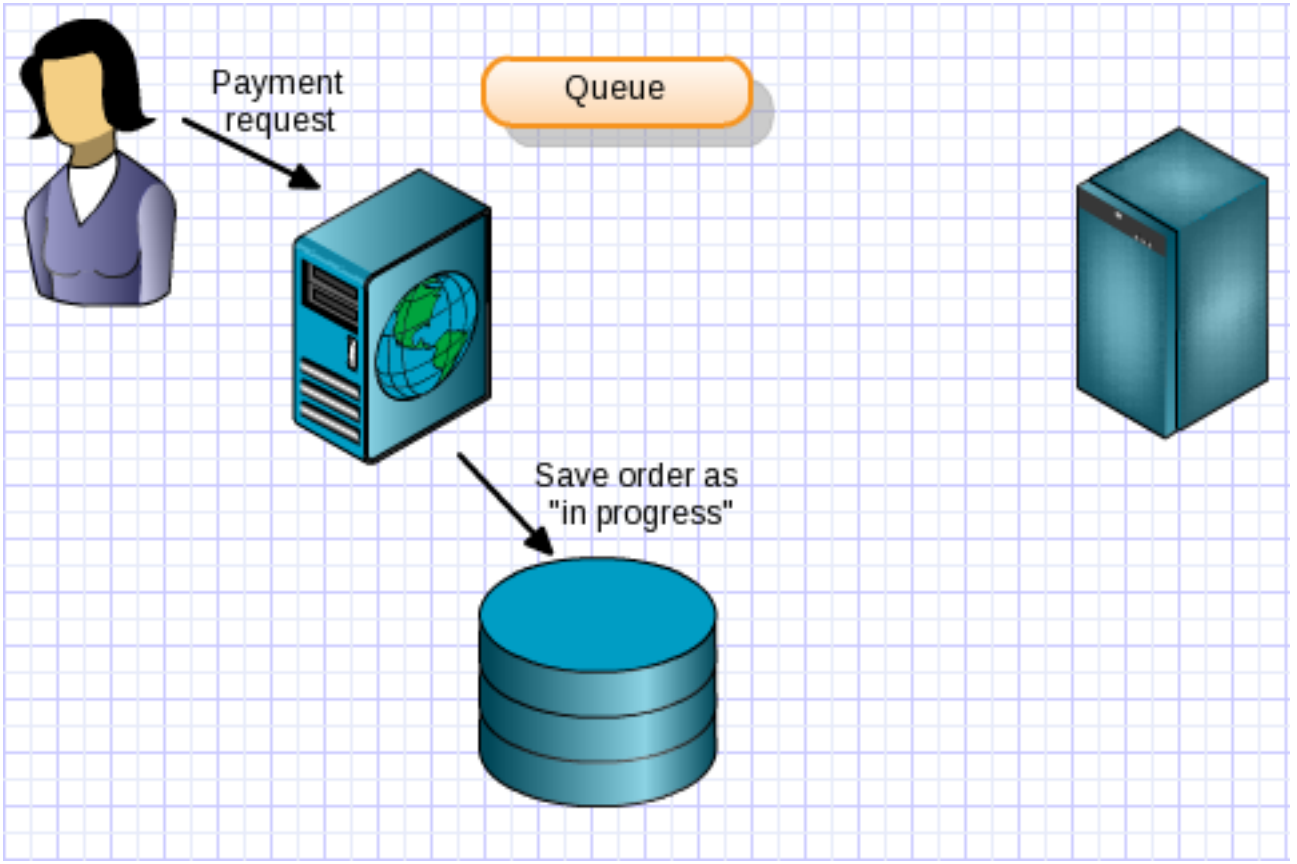
- User closes the window mid-way
- User clicks on refresh
- Web call times out
- CC channels too busy/RPC times out
- Order processing fails after authorisation

On top of that, we're wasting web/db resources!

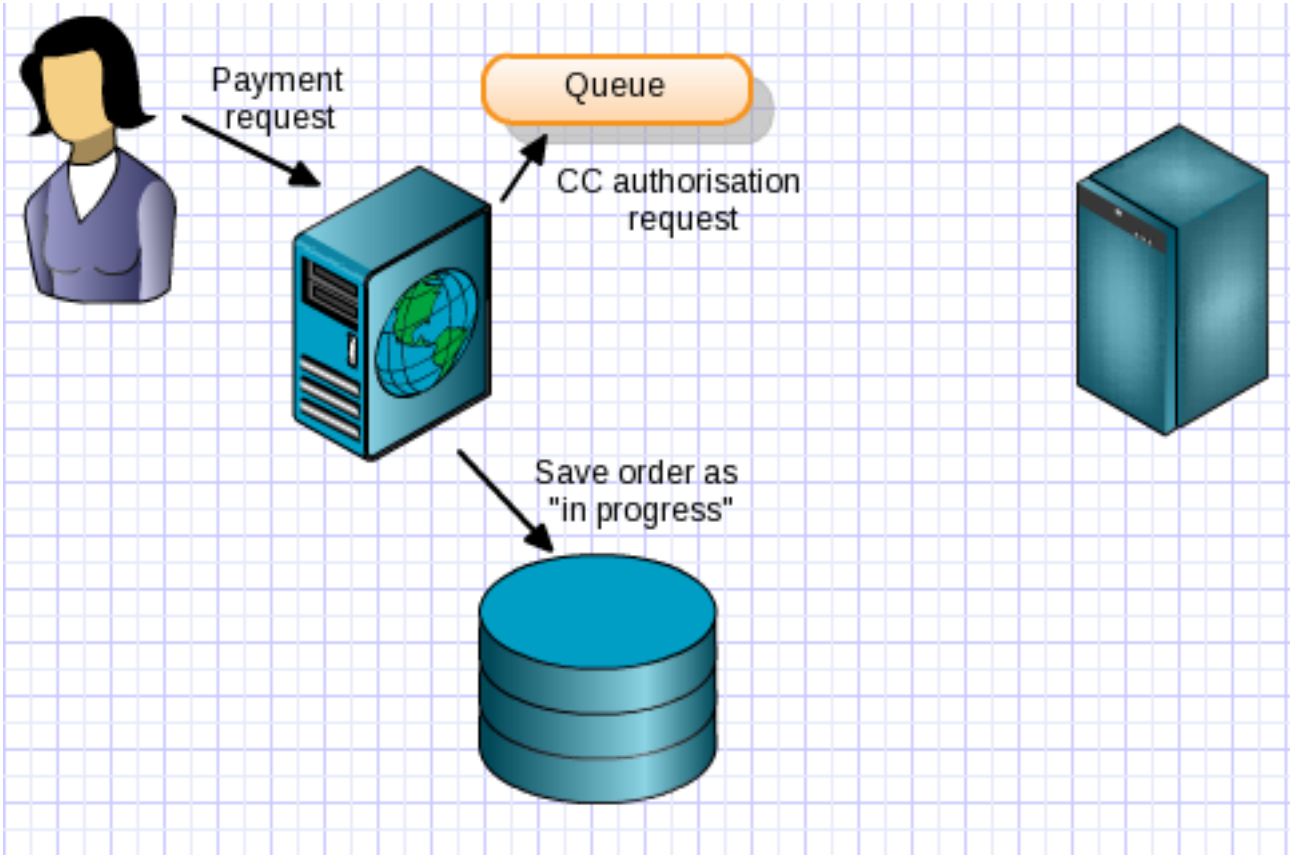
Alternative solution



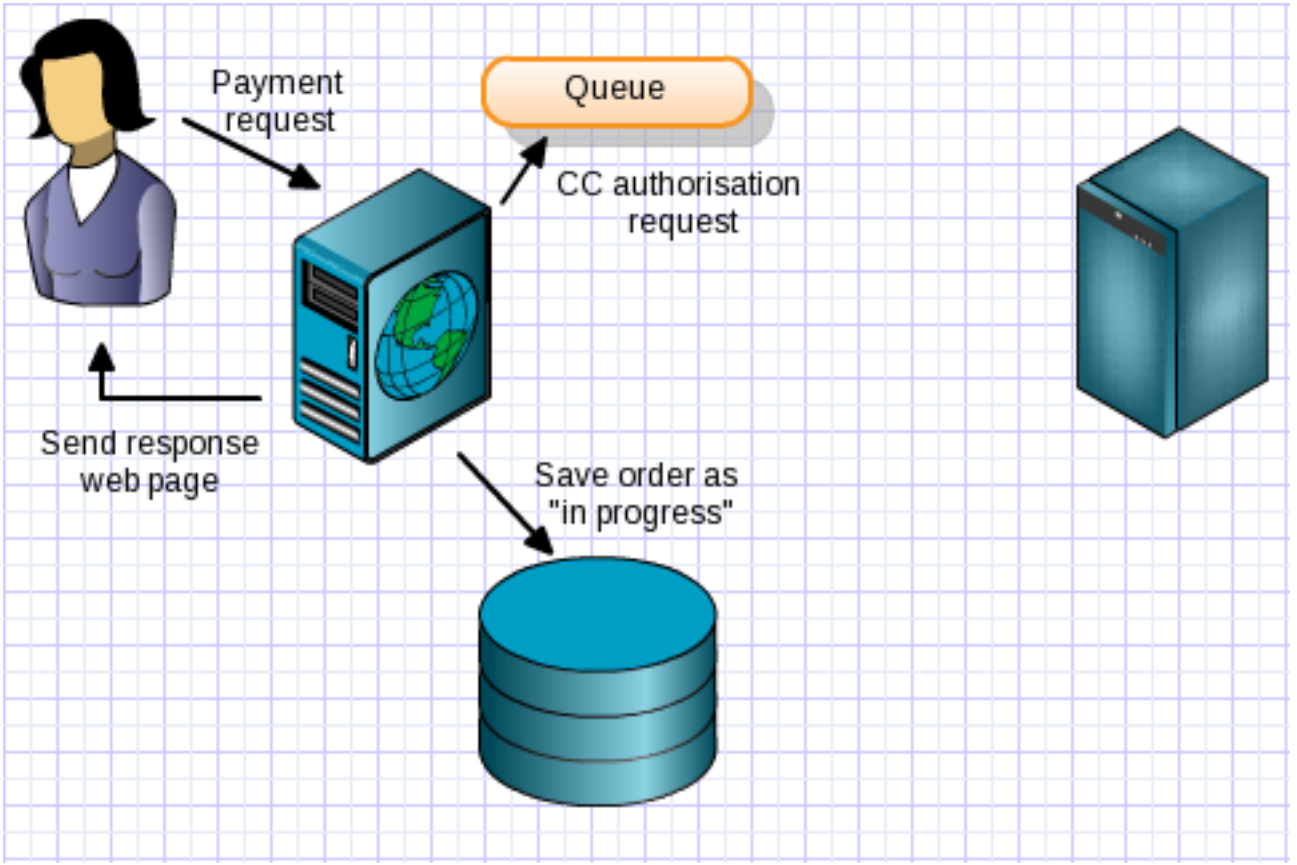
Alternative solution



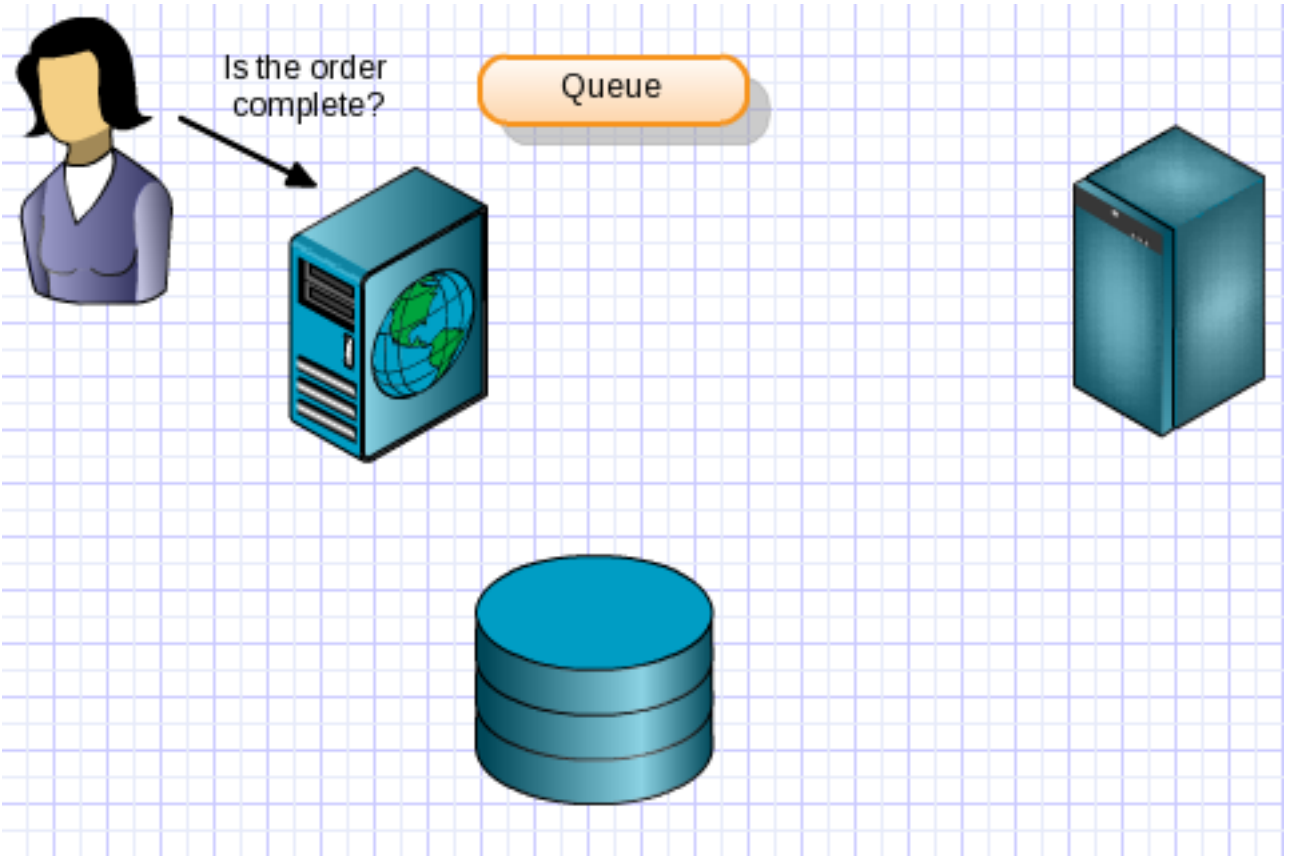
Alternative solution



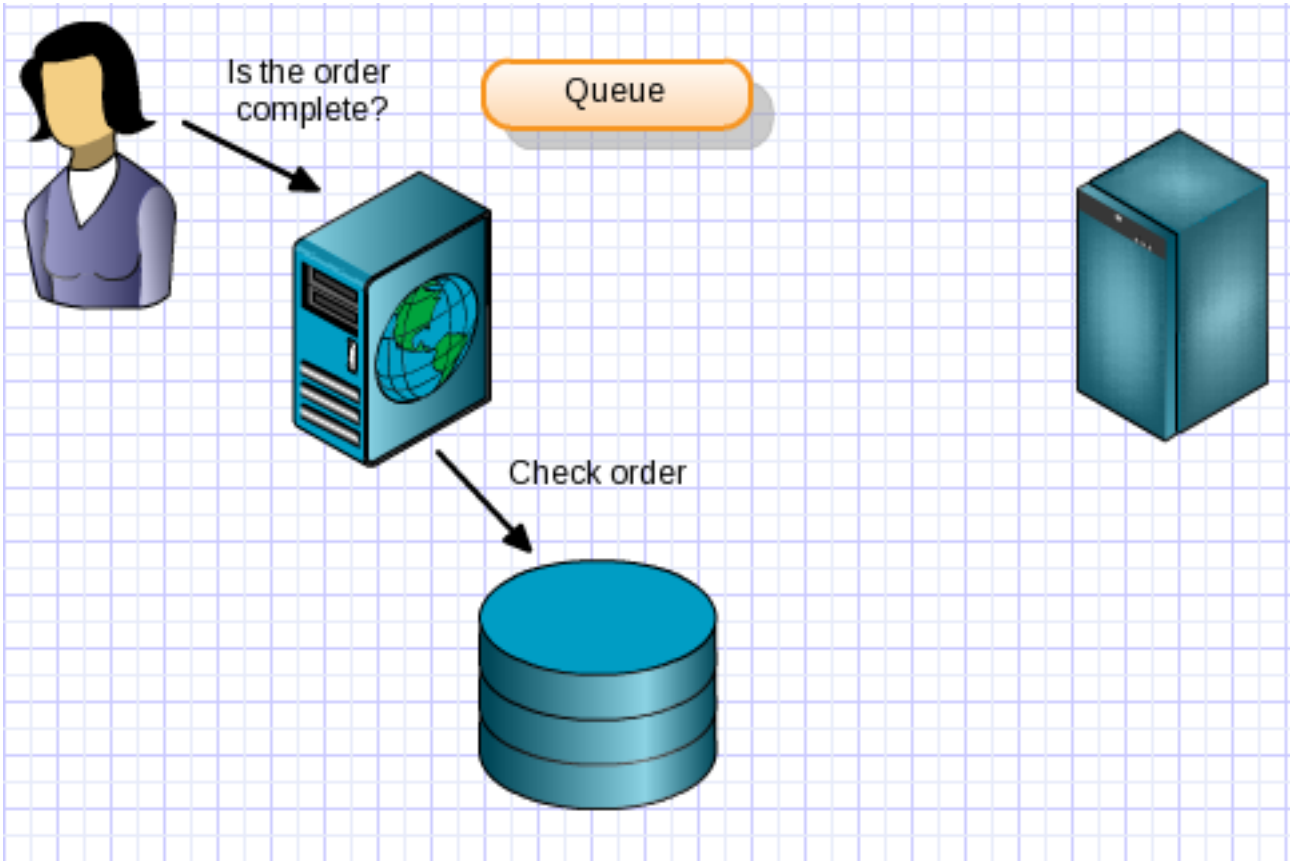
Alternative solution



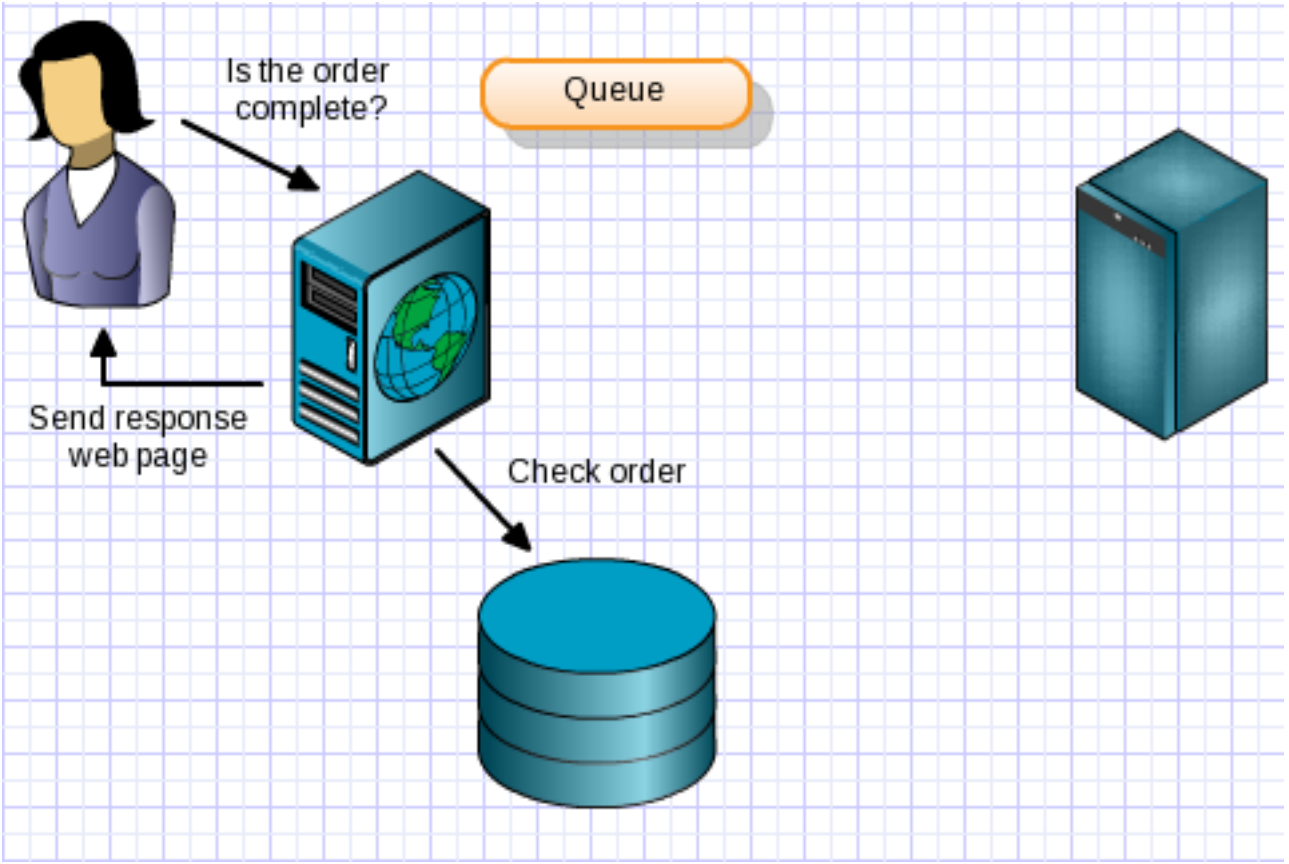
Alternative solution



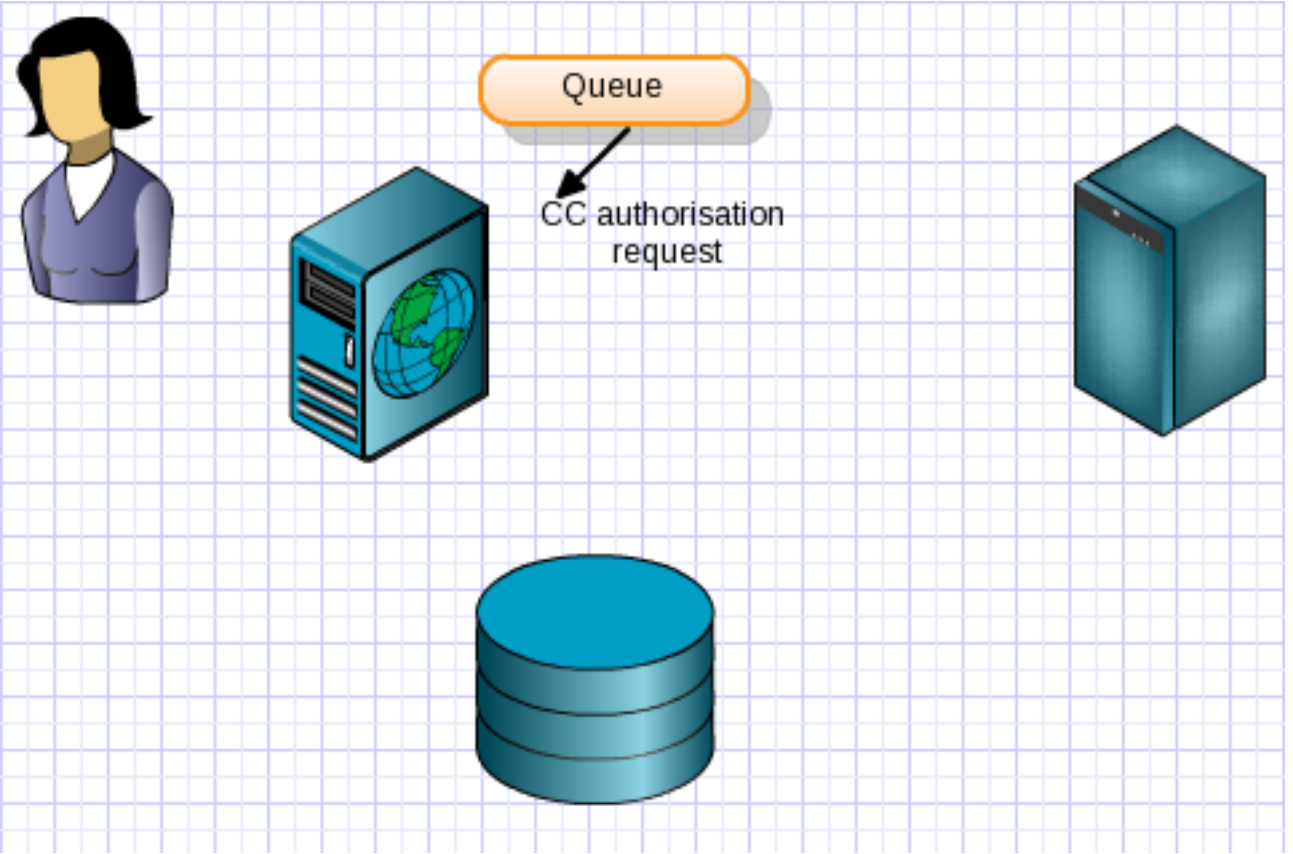
Alternative solution



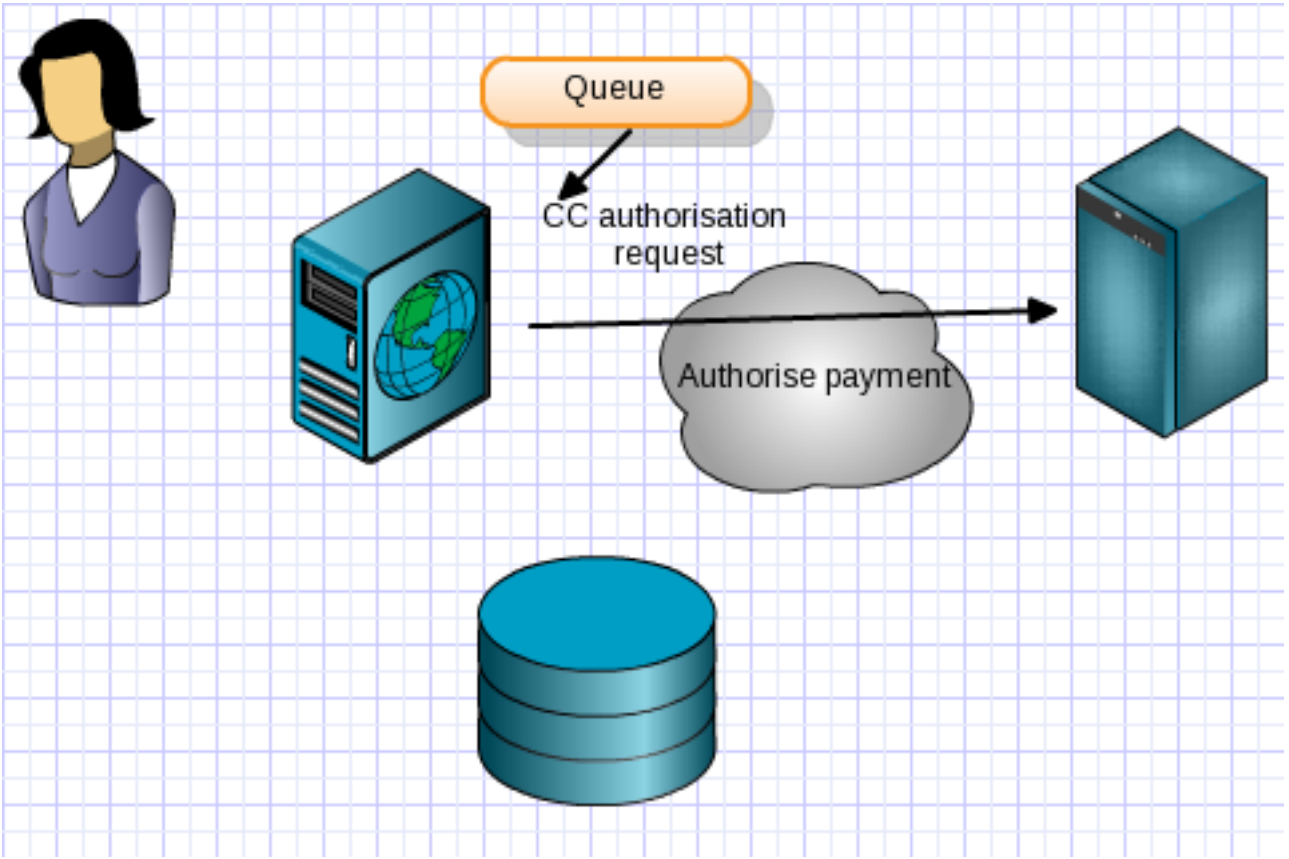
Alternative solution



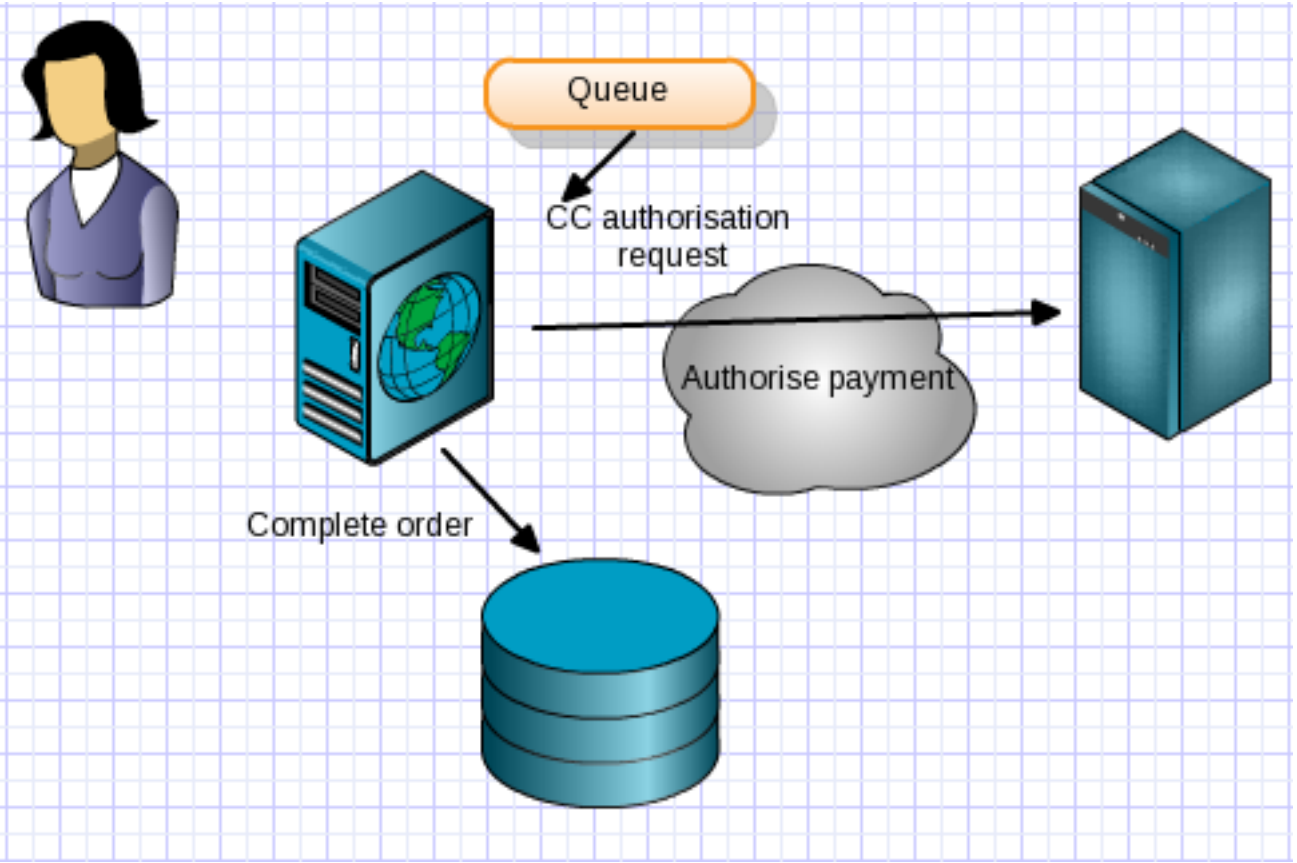
Alternative solution



Alternative solution



Alternative solution



What's improved?

- Closing the window makes no difference
- Refresh makes no difference
- Web call will not time out
- We can wait for CC channels
- Web and DB resources used much better

Some ways to improve this...

- Enqueue operation result, authorise order asynchronously (increase resilience)
- Scale to more servers
- Process cards using dedicated servers (VLAN)
- Avoid polling, send a message to the client

Some other situations where messaging might come in handy

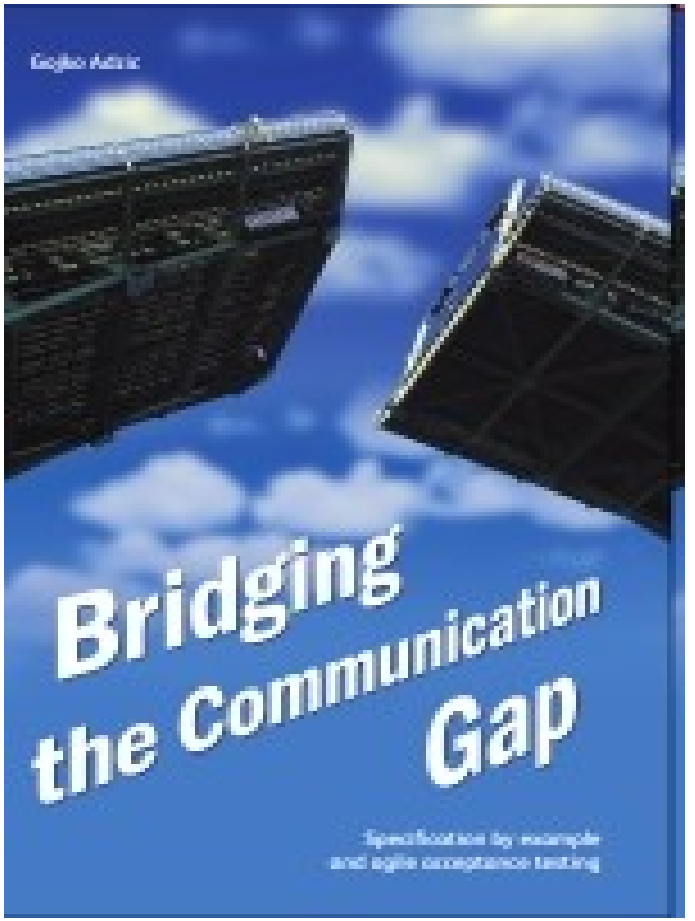
- Distributing work across multiple machines
- Performing a number of actions when something happens (eg notify admin, notify customer)
- Pushing frequent updates to clients

Tools

- ActiveMQ: http://activemq.apache.org/
- NServiceBus: http://www.nservicebus.com/
- AMQP: http://amqp.org

Bridging the Communication Gap

- learn how to improve communication between business people and software implementation teams
- find out how to build a shared and consistent understanding of the domain in your team
- learn how to apply agile acceptance testing to produce software genuinely fit for purpose
- discover how agile acceptance testing affects your work whether you are a programmer, business analyst or a tester
- learn how to build in quality into software projects from the start, rather than control it later



http://www.acceptancetesting.info