# Practical Workshop on Agile Software Development

# PRACTICAL WORKSHOP ON AGILE SOFTWARE DEVELOPMENT

*Biniam Asnake*
*Lead Software Engineer*

*EMAIL biniamasnake@gmail.com*

*LINKEDIN linkedin.com/in/biniama*

*GITHUB github.com/PracticalSoftwareEngineeringWorkshop*

*Please fill this form about your expectation from the workshop*

# HTTP://BIT.LY/AGILE-WORKSHOP-QUESTION

# EXECUTIVE SUMMARY

- The workshop will focus on **practical** experience sharing about **agile** software development methodology and detail day-to-day implementation of one of the agile methodologies i.e. SCRUM as well as step by step **implementation of a real-world of a money-transfer system**.
- At the end of the workshop, trainees will have understanding, experience and knowledge of how American and European IT companies develop applications and be able to develop one using the same methodology and technology stack.

# GOAL

TO INCREASE PRACTICAL KNOWLEDGE OF PROFESSIONAL SOFTWARE DEVELOPMENT IN SHEBA VALLEY

# BINIAM ASNAKE

- **Lead Software Engineer** at bonial.com in Berlin, Germany
- **MSc in Information Science**, Addis Ababa University
- **Java**, Python, **ReactJS**, NodeJS, **Flutter/Dart**, Android …
- **Trainer** of Agile Methodology and Software Engineering
- Blogs on biniamasnake.medium.com

- was **Senior Developer** at Apposit/Paga Tech solutions
- was **Lecturer** at Haramaya University and Lucy University College

# OUTLINE

- *Agile Manifesto and Principles (Theory)*
- *Scrum vs Kanban Methodologies/Frameworks (Theory)*
- Git version control system and *GitHub*
- Project management using *Trello*
- Developing a money transfer application
  - API development using *Spring boot and Java*
  - Front-end app development using *ReactJS (and Javascript)*
  - Mobile app development using *Flutter (and Dart)*
- Continuous Integration and Continuous Delivery (CI/CD - DevOps)
  - *Travis CI*
  - *Heroku* Deployment

“

Organisations which design systems … are constrained to produce designs which are copies of the communication structures of these organisations.

*-Melvin Conway*

“

Team designs are the first draft of your architecture.

*-Michael Nygard (author of RELEASE IT!)*

> In most cases, **a disciplined development process is more important** than effective algorithms, robust architecture, talented engineers, team morale, or open communications.

*eXtremely Distributed Software Development (https://www.xdsd.org/)*

*(Cooke-Davies, 2002; Emam and Koru, 2008; Sauser et al., 2009).*

# WHAT IS AGILE?

THE ABILITY TO CREATE AND RESPOND TO CHANGE IN ORDER TO SUCCEED IN AN UNCERTAIN AND TURBULENT ENVIRONMENT.

*https://www.agilealliance.org/agile101/what-is-agile/*

# WATERFALL MODEL

## What is Agile Software Development?

Agile Software Development is an umbrella term for a **set of methods and practices** based on the values and principles expressed in the **Agile Manifesto**.

Solutions evolve through collaboration between self-organizing, cross-functional teams utilizing the appropriate practices for their context.

*https://www.agilealliance.org/agile101/what-is-agile/*

# A LITTLE BIT OF HISTORY

- In the late 1990's, several methodologies began to gain increasing public attention, each having a different combination of old and new ideas. These methodologies emphasized close collaboration between the development team and business stakeholders; frequent delivery of business value, tight, self-organizing teams; and smart ways to craft, confirm, and deliver code.
- The term "**Agile**" was applied to this collection of methodologies in **early 2001** when **17** software development practitioners gathered in **Snowbird, Utah** to discuss their shared ideas and various approaches to software development. This joint collection of values and principles was expressed in the **Manifesto for Agile Software Development** and the corresponding **twelve principles.**
- The Agile Alliance was formed shortly after this gathering to encourage practitioners to further explore and share ideas and experiences.

# AGILE MANIFESTO

*www.agilemanifesto.org*

“We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value: …

*www.agilemanifesto.org*

“

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

*www.agilemanifesto.org*

“

That is, while there is value in the items on the right, we value the items on the left more.

*www.agilemanifesto.org*

# PRINCIPLES BEHIND THE AGILE MANIFESTO

# WE FOLLOW THESE PRINCIPLES:

- Our highest priority is to satisfy the customer through early and **continuous delivery of valuable software**.
- **Welcome changing requirements**, even late in development. Agile processes harness change for the customer's competitive advantage.
- **Deliver** working software **frequently**, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
  - *Quiz*

    *How often do you think Amazon deploys to production/live server?*

# EVERY 11.7 SECONDS

Google

amazon release frequency

All | Images | News | Videos | More | Settings | Tools

About 54,600,000 results (0.51 seconds)

After moving to its own cloud, **Amazon** engineers deploy code every 11.7 seconds, on average—reducing both the number and duration of outages at the same time. Netflix engineers deploy code thousands of times per day. How about your team? Feb 4, 2016

blog.newrelic.com › technology › data-culture-survey-res...

Most Companies Deploying Code Weekly, Daily, or Hourly

About featured snippets • Feedback

# WE FOLLOW THESE PRINCIPLES:

- Business people and developers must **work together** daily throughout the project.
- Build projects around **motivated individuals**. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is **face-to-face conversation**.
- **Working software** is the primary measure of progress.
- Agile processes promote **sustainable development**. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

# WE FOLLOW THESE PRINCIPLES:

- Continuous attention to **technical excellence and good design** enhances agility.
- **Simplicity**--the art of maximizing the amount of work not done--is essential.
- The best architectures, requirements, and designs emerge from **self-organizing teams.**
- At regular intervals, **the team reflects** on how to become more effective, then tunes and adjusts its behavior accordingly.

# SCRUM FRAMEWORK

# SCRUM IN NUTSHELL

- Split your organization into small, cross-functional, self-organizing teams.

- Split your work into a list of small, concrete deliverables.
  - Sort the list by priority and estimate the relative effort of each item.

# … SCRUM IN NUTSHELL

- Split time into short fixed-length iterations (usually 1 – 4 weeks), with potentially shippable code demonstrated after each iteration.

- Optimize the release plan and update priorities in collaboration with the customer, based on insights gained by inspecting the release after each iteration.
- Optimize the process by having a retrospective after each iteration.

“

Instead of a **large group** spending a **long time** building a **big thing**, we have a **small team** spending a **short time** building a **small thing**. But **integrating** regularly to see the whole.

- In Ken Schwaber's words, Scrum is **not a methodology**, it is a *framework*.
  - What that means is that Scrum is **not really going to tell you exactly what to do.**
- The strength and pain of Scrum is that you are forced to adapt it to your specific situation.

# KANBAN IN NUTSHELL

- **Visualize the workflow**
  - ***Split the work into pieces***, write each item on a card and put on the wall.
  - ***Use named columns*** to illustrate where each item is in the workflow.
- **Limit Work In Progress (WIP)** – assign explicit limits to how many items may be in progress at each workflow state.
- **Measure the lead time** (average time to complete one item, sometimes called "cycle time"), optimize the process to make lead time as small and predictable as possible.

# SCRUM VS KANBAN

Both Scrum and Kanban are based on incremental development,

i.e. break the work into smaller pieces.

# SCRUM VS KANBAN

**Compare tools for understanding, not judgment.**

*Knife or fork – which tool is better?*

**Scrum and Kanban are neither perfect nor complete.**

They **don't tell you everything** that you need to do, they just **provide certain constraints & guidelines.**

**The value of a tool is that it limits your options.**

E.g.: **Scrum** constrains you to have ***timeboxed iterations*** and ***cross-functional teams***, and

**Kanban** constrains you to use ***visible boards*** and ***limit the size of your queues***.

➤ Mix and match the tools as you need!

# ROLES IN SCRUM

**Scrum prescribes 3 roles:**

1. Product Owner (sets product vision & priorities),
2. Team (implements the product) and
3. Scrum Master (removes impediments and provides process leadership).

*Kanban doesn't prescribe any roles at all.* ***Means****: you can but you don't have to.*

**Additional Roles?**

In both Scrum and Kanban, you are **free to add** whatever additional roles you need.

**Be careful!** when adding roles though, make sure the additional roles actually add value and don't conflict with other elements of the process.

- The general mindset in both Scrum and Kanban is "**less is more**". So when in doubt, start with less.

# SCRUM: TIMEBOXED ITERATIONS

A Scrum team will **only commit** to items that they can **complete within one iteration** (based on the definition of "Done").

**Team #1 (single cadence)**

*"We do Scrum iterations"*

**Team #2 (three cadences)**

Everyweek: release

Every 2 weeks: planning

Every 4 weeks: retro

**Team #3 (mostly event-driven)**

Release: whenever there is MMFs

Planning: when run out of stuff to do

Retro: every 4 weeks

➤ Note: retrospective meeting to tweak and improve our process

# SCRUM VS KANBAN BOARDS

Scrum limits WIP per iteration

Kanban limits WIP per workflow state

*(general idea is to*

*limit WIP of all workflow states)*

- Every iteration in SCRUM is called **Sprint**
- Scrum teams measure **velocity** – how many items (or corresponding units such as "story points") get done per iteration.
- Kanban teams measure and predict **lead time**, i.e. the average time for an item to move all the way across the board.
  - Having predictable lead times allows us to commit to **SLAs** (service-level agreements) and make realistic release plans.
    - Too low kanban limit => idle people => bad productivity
    - Too high kanban limit => idle tasks => bad lead time

# SCRUM AND KANBAN ARE EMPIRICAL

- You are expected to **experiment** with the process and customize it to your environment.
- Neither Scrum nor Kanban provide all the answers – they just give you a basic **set of constraints** to drive your own process improvement.

- The most critical element of this is the **feedback loop**.

  ***Change something => Find out how it went => Learn from it => Change something again.***

- You want as short a feedback loop as possible, so you can adapt your process quickly.

# THE BOARDS OVER TIME

A **Scrum board** is owned by exactly one Scrum team.

- A Scrum team is **cross-functional**, it contains all the skills needed to complete all the items in the iteration.

In **Kanban**, cross-functional teams are optional, and a board ***doesn't need to be owned by one specific team.***

- A board is related to one workflow, not necessarily one team.

# SCRUM PRESCRIBES ESTIMATION AND VELOCITY

Scrum teams estimate the relative size (= amount of work) of each item that they commit to.

Velocity is a measure of capacity – how much stuff we can deliver per sprint.

*By adding up the size of each item completed at the end of each sprint, we get velocity.*

Knowing the average velocity is nice, because then we can make realistic predictions about which items we can complete in upcoming sprints, and therefore make realistic release plans.

# KANBAN DOESN'T PRESCRIBES ESTIMATION AND VELOCITY

**In Kanban, estimation is not prescribed**. Hence,

1. Some teams choose to make estimates and measure velocity just like in Scrum.
2. Other teams break items into pieces of roughly the same size and measure velocity simply in terms of how many items were completed per unit of time (for example features per week).
3. Some teams group items into MMFs and measure the average lead time per MMF, and use that to establish Service-Level Agreements (SLAs).

   *E.g.: "when we commit to an MMF it will always be delivered within 15 days".*

# SCRUM AND KANBAN ALLOWS WORKING ON MULTIPLE PRODUCTS SIMULTANEOUSLY

**1. team focus on one product per sprint**

**2. work on features from both products each sprint:**

# SCRUM AND KANBAN ALLOWS WORKING ON MULTIPLE PRODUCTS SIMULTANEOUSLY

**1. coloured cards**

**2. swimlanes**

# SCRUM PRESCRIBES PRIORITISED PRODUCT BACKLOG

In Scrum, **prioritisation** is always done by sorting the product backlog, and *changes to priorities take effect in the next sprint* (not the current sprint).

In Kanban you can choose **any** prioritisation scheme (or even **none**), and *changes take effect as soon as capacity becomes available* (rather than at fixed times).

- There ***may or may not be a product backlog***, and it may or may not be prioritised.
- The most left column typically fulfills the same purpose as a Scrum product backlog.

# SCRUM PRESCRIBES DAILY STANDUP

- A Scrum team has a short meeting (at most 15 minutes) every day at the same time & same place.
- The purpose of the meeting is to spread information about:
  - what is going on,
  - plan the current day's work, and
  - identify any significant problems.
  - it is usually done standing (to keep it short & maintain a high energy level).
- Daily standups are not prescribed in Kanban, but most Kanban teams seem to do it anyway. It's a great technique, regardless of which process you use.

*In Scrum the format of the meeting is **people-oriented** - every person reports one by one.*

*Many Kanban teams use a more **board-oriented** format, focusing on bottlenecks and other visible problems.*

# SCRUM: BURNDOWN CHARTS

- A sprint burndown chart shows, on a daily basis, **how much work remains in the current iteration.**
- In Scrum, sprint burndown charts are used as one of the **primary tools** for tracking the progress of an iteration.
- The main purpose of a burndown chart is to easily find out as early as possible if we are behind or ahead of schedule, so that we can adapt.

# KANBAN CHARTS

In Kanban, **no particular type of chart is prescribed.** But you are of course allowed to use any type of chart you like (including burndowns).

Here's an example of a Cumulative Flow diagram. This type of chart illustrates nicely how smooth your flow is and how WIP affects your lead time.

Here's how it works. Every day, total up the number of items in each column on the Kanban board and stack these on the Y axis. So on day 4 there were 9 items in the board. Starting from the right-most column there was 1 item in Production, 1 item in Test, 2 items in Dev, and 5 items in the Backlog. If we plot these points every day and connect the dots we get a nice diagram like the one above. The vertical and horizontal arrows illustrate the relationship between WIP and lead time.

The horizontal arrow shows us that items added to the backlog on day 4 took on average 6 days to reach production. About half of that time was Test. We can see that if were to limit the WIP in Test and Backlog we would significantly reduce the total lead time.

# SUMMARY: SIMILARITIES

- Both are Lean and Agile.
- Both use pull scheduling.
- Both limit WIP.
- Both use transparency to drive process improvement.
- Both focus on delivering releasable software early and often.
- Both are based on self-organizing teams.
- Both require breaking the work into pieces.
- In both, the release plan is continuously optimized based on empirical data (velocity / lead time).

# SUMMARY: DIFFERENCES

| Scrum | Kanban |
|---|---|
| Timeboxed iterations prescribed. | Timeboxed iterations optional.<br>Can have separate cadences for planning, release, and process improvement. Can be event- driven instead of timeboxed. |
| Team commits to a specific amount of work for this iteration. | Commitment optional. |
| Uses Velocity as default metric for planning and process improvement. | Uses Lead time as default metric for planning and process improvement. |
| Cross-functional teams prescribed. | Cross-functional teams optional.<br>Specialist teams allowed. |
| Items must be broken down so they can be completed within 1 sprint. | No particular item size is prescribed. |
| Burndown chart prescribed | No particular type of diagram is prescribed |
| WIP limited indirectly (per sprint) | WIP limited directly (per workflow state) |
| Estimation prescribed | Estimation optional |
| Cannot add items to ongoing iteration. | Can add new items whenever capacity is available |
| A sprint backlog is owned by one specific team | A kanban board may be shared by multiple teams or individuals |
| Prescribes 3 roles (PO/SM/Team) | Doesn’t prescribe any roles |
| A Scrum board is reset between each sprint | A kanban board is persistent |
| Prescribes a prioritized product backlog | Prioritization is optional. |

# //TODO

*what are we going to do today in the workshop?*

# TASKS

- Setup cross-functional team(s)
  - **Developers** (backend, frontend, mobile, api)
  - Database Developers/Engineers
  - UI/UX Designers
  - **Testers** (manual, automated test engineers/software engineers in Test)
  - DevOps (Developers plus Operations)
  - Architects (software arc., database arc., system arc., enterprise arc., integration arc. …)
  - **Others:** ***Product Owner, Scrum master****, Team Lead, Lead Developer*
- Product Kickoff Meeting
  - Presentation of the product we are going to develop by the **Product Owner** to the team

# TASKS

- Setup Tools
  - Project management tools: **Trello**, JIRA, YouTrack, …
  - Source code repositories: **GitHub**, GitLab, BitBucket
  - Communication tools: **Slack,** HipChat, Skype for Business, …
- Product feature identification and Breakdown (***@Team***)
- Designing
  - Architecture design
  - Database design
  - User Interface design
- Technology comparison and selection (***@Team***)
- Sprint Planning and Estimation meeting (***@scrum master*** *and* ***@team***)
- *Coding Time! yeah!* (***@Team***)
- Sprint Review (***@Team***)
- Sprint Retrospective (***@scrum master*** *and* ***@team***)

# SETUP CROSS-FUNCTIONAL TEAM(S)

FIRST EXERCISE AS A TEAM
1. GIVE IT A NAME
2. APPOINT A PRODUCT OWNER

# THE 3 MAIN SCRUM ROLES:

# TEAM, PRODUCT OWNER AND SCRUM MASTER

# SELF-ORGANISE A CROSS-FUNCTIONAL TEAM

# PRODUCT KICKOFF

# TRY TO UNDERSTAND THE CLIENT REQUIREMENT AND MAKE SURE THE DELIVERABLE FITS THE REQUIREMENT

# MAKE THE CLIENT HAPPY AT ALL PHASES OF DELIVERY

# ALWAYS DEVELOP CLEAN ARCHITECTURE AND CODE

# EXERCISE:
# CLIENT EXPLAINS THE PRODUCT FEATURES TO THE PRODUCT OWNERS (PO)
# THEN, PO EXPLAINS THE FEATURES TO THE TEAM

[IN THE MEANTIME, THE TEAM CAN CREATE ACCOUNT ON TRELLO.COM]

# PRODUCT FEATURE IDENTIFICATION AND BREAKDOWN

# EVENT STORMING (WWW.EVENTSTORMING.COM/)

- EventStorming is a **flexible workshop format for collaborative exploration of complex business domains.**
- It comes in different flavours, that can be used in different scenarios:
  - to assess health of an existing line of business and to discover the most effective areas for improvements;
  - to explore the viability of a new startup business model;
  - to envision new services, that maximise positive outcomes to every party involved;
  - to design clean and maintainable Event-Driven software, to support rapidly evolving businesses

# INSTEAD OF WORKING ON ONE COMPONENT AFTER ANOTHER, TRY DELIVERING VALUE BY DEVELOPING USABLE AND UTTERABLE COMPONENTS

## EXAMPLE

*Read more on*

*https://jimmybogard.com/vertical-slice-architecture/*

*https://www.ghyston.com/insights/architecting-for-maintainability-through-vertical-slices*

# YOUR WORK: FEATURE IDENTIFICATION AND BREAKDOWN

## ALSO CALLED – USER STORY MAPPING

*Steps:*

1. *Identify the actors in the system*
2. *Identify features that can be released at once using the MVP principle*
3. *Break each feature down into items that can be given to a developer to work on*

*Read more on*

*https://plan.io/blog/user-story-mapping/*

# FEATURE DESCRIPTION TEMPLATE

1. *Title of the feature*
2. *Contact Person*
   - *Who should the developer talk to if he/she has a question?*
3. *Description/Use case*
   - *Detail description of what is required with Mockups and sample data*
   - *E.g. As a user, I would like to create a money transfer account.*
     - *Add required fields such as firstName, lastName, …*
     - *Mention Business logic e.g. duplicate phone number is not allowed.*
4. *Acceptance Criteria*

*Things that are important to consider this task as DONE*

- *E.g. When I enter all the required information, I should get a message with "Account created"*
- *E.g. If I try to register a user twice, I should get an error "Cannot create a duplicate user"*

# EXERCISE

## SETUP PROJECT MANAGEMENT TOOL

## CREATE AN ACCOUNT ON TRELLO.COM AND CREATE A BOARD

## INVITE ME TO YOUR BOARD @BINIAMA

# TRELLO BOARD FOR MONEY TRANSFER APPLICATION – JUST STARTING

# TRELLO CARD – CREATE ACCOUNT

# TRELLO BOARD – FULL VIEW

# PRODUCT SPECIFICATION

- We need a money transfer application that works on both web and mobile platforms.
- The actors of the system are **Users** who send money to each other and **Admins** who can manage the whole system
- The application data should be stored in a persistence store/database.
- Business Rules
  - Duplicate accounts are not allowed
  - Once committed, a transaction cannot be updated.
- The application should be able to perform the following tasks:
  - Create user Account/ Register
  - List all accounts (*for admin*)
  - Show individual account/ User Profile
  - Update account/ Update password, …
  - Delete account
  - Create Transfer/ Make money transfer
  - List all transfers/ Transaction History
  - Show individual transfer
  - Delete transfer

# TECHNOLOGY COMPARISON AND SELECTION

*You can find the template for the matrix here*

*https://docs.google.com/document/d/1WEzmjfwH4xD5tcmg8jjUu7FpcjJscL3CsU8y_k7eiwg/edit?usp=sharing*

# AVOID SUBJECTIVITY AND MAKE YOUR DECISION BASED ON FACTS/DATA

## ’TO A MAN WITH A HAMMER, EVERYTHING LOOKS LIKE A NAIL’ (ABRAHAM MASLOW)

GET DEEPER KNOWLEDGE OF THE STUFF YOU ALREADY KNOW

GET FAMILIAR WITH THE STUFF YOU KNOW YOU DON'T KNOW

# technical knowledge

# KNOWLEDGE: I-SHAPED, T-SHAPED, PI-SHAPED AND COMB-SHAPED

# COMPARE THE TECHNOLOGIES/FRAMEWORKS USING MULTIPLE CRITERIA

- Comparison methods -

  **USE ONLY ONE METHOD**

  - ***Numbers***
    - e.g. 1 up to 5

  (5 is best and 1 is worst)

  - ***Yes or No***
  - ***few Thumbs up or Down***
- ***Assign weight to each criteria for unbiased judgement***
- **DON'T BE BAISED!!!**

# TECHNOLOGY COMPARISON MATRIX

*Technology is a tool and we should use Objective methods to select the best technology and tool for the job at hand.*

*We will use a comparison method of numbers from 1 to 5, where 1 is the worst and 5 is the best.*

*We will use weighting (assigning value to be multiplied by) so that we can prioritize the most important criteria.*

| Criteria | Weight (out of 5) | Web API development frameworks | | | | |
|---|---|---|---|---|---|---|
| | | ExpressJS/ Node JS | Spring Boot/ Java | ASP.Net/ C# | Django/ Python | Laravel / PHP |
| Free and open source | 3 | 5 | 5 | 2 | 5 | 4 |
| Performance/Speed | 4 | 4 | 4 | 2 | 3 | 1 |
| Stability (version 0.1 vs version 10) | 4 | 3 | 5 | 5 | 4 | 3 |
| Learning curve (easy or difficult) | 5 | 3 | 4 | 3 | 4 | 2 |
| Documentation and community support (on stackoverflow.com and other channels) | 3 | 4 | 5 | 3 | 3 | 2 |
| Flexible framework vs Opinionated | 1 | 5 | 4 | 5 | 1 | 3 |
| Developer availability (for hiring new developers) | 3 | 3 | 3 | 4 | 2 | 3 |
| Free hosting | 4 | 4 | 4 | 1 | 3 | 1 |

*Then, multiply the weight with the value, get the total value and you have the chosen framework.*

*NOTE: You should also consider your team's preference to work with these technologies.*

# TECHNOLOGY COMPARISON MATRIX

| **Criteria** | **Weight** (out of 5) | **Web API development frameworks** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Express JS/ Node JS | ExpressJS **multiplied by Weight** | Spring Boot/ Java | SpringBoot **multiplied by Weight** | ASP.Net/ C# | ASP.Net **multiplied by Weight** | Django/ Python | Django **multiplied by Weight** | Laravel/ PHP | Laravel **multiplied by Weight** |
| Free and open source | **3** | 5 | *15* | 5 | *15* | 2 | *6* | 5 | | 4 | |
| Performance/Speed | **4** | 4 | *16* | 4 | *16* | 2 | *8* | 3 | | 1 | |
| Stability (version 0.1 vs version 10) | **4** | 3 | *12* | 5 | *20* | 5 | *20* | 4 | | 3 | |
| Learning curve (easy or difficult) | **5** | 3 | *15* | 4 | *20* | 3 | *15* | 4 | | 2 | |
| Documentation and community support | **3** | 4 | *12* | 5 | *15* | 3 | *9* | 3 | | 2 | |
| Flexible framework vs Opinionated | **1** | 5 | *5* | 4 | *4* | 5 | *5* | 1 | | 3 | |
| Developer availability (for hiring new developers) | **3** | 3 | 9 | 3 | 9 | 4 | 12 | 2 | | 3 | |
| Free hosting | **4** | 4 | 16 | 4 | 16 | 1 | 4 | 3 | | 1 | |
| **TOTAL** | **-** | **-** | **100** | | **115** | | **77** | | | | |

# SCRUM IN PRACTICE

# A TYPICAL SCRUM PROCESS FLOW

# FOLLOW SCRUM AND CONDUCT THE CEREMONIES

## SCRUM CEREMONIES in a SPRINT

1. sprint planning and estimation
   - ***using fibonacci series (1, 2, 3, 5, 8, 13)***
   - ***Pocker cards***
2. daily standup
   - *What did you do yesterday?*
   - *What did you do today?*
   - *Any blockers or impediments?*
3. sprint review (for the team + PO)
4. sprint demo (for stakeholders)
5. Sprint retrospective
   - *The Good, The Bad, To Improve, To Start*

# SPRINT PLANNING AND ESTIMATION MEETING

# EXERCISE

## SPECIFY THE DETAILS OF "CREATE ACCOUNT"

## ESTIMATE THE EFFORT OF TASK USING FIBONACCI SERIES

1 2 3 5 8 13

# TIME FOR

ARCHITECTURE DESIGN

DATABASE DESIGN

USER INTERFACE DESIGN

# USE C4MODEL.COM FOR DESIGNING ARCHITECTURAL DIAGRAMS

## Visualising software architecture

1 As a team, agree upon a set of abstractions you will use to communicate software architecture. The "C4 model" is a hierarchical way to think about the static structures of a software system in terms of containers, components and classes (or code):

> A **software system** is made up of one or more **containers** (web applications, mobile apps, desktop applications, databases, file systems, etc), each of which contains one or more **components**, which in turn are implemented by one or more **classes** (or **code**).

2 Visualise this hierarchy by creating a collection of System Context, Container, Component and (optionally) UML class diagrams. Think about these diagrams as maps of your software, showing different levels of detail.

[System Context] techtribes.je

[Containers] techtribes.je

[Components] techtribes.je - Content Updater

### Level 1: System Context

A System Context diagram is a good starting point for diagramming and documenting a software system, allowing you to step back and see the big picture. Draw a diagram showing your system as a box in the centre, surrounded by its users and the other systems that it interacts with. Detail isn't important here as this is your zoomed out view showing a big picture of the system landscape. The focus should be on people (actors, roles, personas, etc) and software systems rather than technologies, protocols and other low-level details.

### Level 2: Containers

The next step is to illustrate the high-level technology choices with a Container diagram. A "container" is something like a web application, mobile app, desktop application, database, file system, etc. Essentially, a container is a separately deployable unit that executes code or stores data. The Container diagram shows the high-level shape of the software architecture and how responsibilities are distributed across it. It also shows the major technology choices and how the containers communicate with one another.

### Level 3: Components

Next you can zoom in to each container further to visualise the major structural building blocks and their interactions. The Component diagram shows how a container is made up of a number of components, what each of those components are, their responsibilities and the technology/implementation details. If your components don't all fit on a single diagram, create multiple versions showing different portions of the container.

3 A common set of abstractions is more important than a common notation, but do ensure that your notation (shapes, colours, line styles, acronyms, etc) makes sense. If in doubt, add a diagram key/legend, even when using UML.

4 Use the elements in your model of the static structure to create additional supplementary diagrams in order to communicate runtime behaviour and deployment (the mapping of containers to infrastructure).

"**Software Architecture for Developers**"
A developer-friendly, practical and pragmatic guide to lightweight software architecture, technical leadership and the balance with agility.

Structurizr
Visualise, document and explore your software architecture

A collection of tooling to help you visualise, document and explore your software architecture.

@simonbrown

# USE C4MODEL.COM FOR DESIGNING ARCHITECTURAL DIAGRAMS

[System Context] techtribes.je

[Containers] techtribes.je

[Components] techtribes.je - Content Updater

# C4 CORE DIAGRAMS (WWW.INFOQ.COM/ARTICLES/C4-ARCHITECTURE-MODEL/)

**Level 1: System Context diagram**

- A System Context diagram is a good starting point for diagramming and documenting a software system, allowing you to step back and see the big picture.
- Draw a diagram showing your system as a box in the centre, surrounded by its users and the other systems that it interacts with.

**Level 2: Container diagram**

- Once you understand how your system fits in to the overall IT environment, a really useful next step is to zoom-in to the system boundary with a Container diagram.
- A "container" is something like a server-side web application, single-page application, desktop application, mobile app, database schema, file system, etc. Essentially, a container is a separately runnable/deployable unit (e.g. a separate process space) that executes code or stores data.
- The Container diagram shows the high-level shape of the software architecture and how responsibilities are distributed across it.
- It also shows the major technology choices and how the containers communicate with one another.
- It's a simple, high-level technology focussed diagram that is useful for software developers and support/operations staff alike.

# C4 CORE DIAGRAMS (WWW.INFOQ.COM/ARTICLES/C4-ARCHITECTURE-MODEL/)

**Level 3: Component diagram**

- Next you can zoom in and decompose each container further to identify the major structural building blocks and their interactions.
- The Component diagram shows how a container is made up of a number of "components", what each of those components are, their responsibilities and the technology/ implementation details.

**Level 4: Code diagram**

- Finally, you can zoom in to each component to show how it is implemented as code; using UML class diagrams, entity relationship diagrams or similar.
- This is an **optional** level of detail and is often available on-demand from tooling such as IDEs.
- This level of detail is not recommended for anything but the most important or complex components.

# PRINCIPLES OF MICROSERVICES ARCHITECTURE

## BY
## SAM NEWMAN

(THE NEXT SLIDE CONTAINS A LINK TO A VIDEO FILE. PLEASE WATCH)

DEVOXX PRESENTS

# 20 YEARS OF JAVA
## JUST THE BEGINNING

# 7 PRINCIPLES OF MICROSERVICES

1. Model around business concepts.
2. Adopt a culture of automation.
3. Hide internal implementation details.
4. Decentralize all the things.
5. Independently deployable.
6. Isolate failure.
7. Highly observable.

***Source*** *https://medium.com/yourapi/7-principles-of-microservices-made-developing-easy-for-us-60c0e32e7c86*

# THE ARCHITECTURE FOR THE MONEY TRANSFER APPLICATION

# THE ARCHITECTURE FOR THE MONEY TRANSFER APPLICATION

# Business Rules

1 account can make multiple/many transfers

1 transfer can be made by only 1 sender account and 1 receiver account

# UI DESIGN REGISTRATION

# INTRO TO GIT

*https://github.com/biniama/git-101*

# FIRST-TIME GIT SETUP

You need to run the following commands to configure your user name and email to use Git.

```
git config --global user.name "John Doe"
git config --global user.email johndoe@example.com
```

# EXERCISE

## GO TO HTTPS://GITHUB.COM/BINIAMA/GIT-101-ACT

```
git clone https://github.com/biniama/git-101-act
```

*Change the content of the file to "Hello, <your name>!"*

```
git status
git add .
git commit -m "message"
git push
```

# FOLLOW GIT WORKFLOW

[IT MIGHT SEEM COMPLEX BUT ONCE YOU LEARN HOW TO DO IT, IT MAKES YOUR LIFE EASIER]

# SPRINT REVIEW

*This is a presentation of all the works that was done in the sprint to THE TEAM AND THE PRODUCT OWNER.*

## *Guideline to run the meeting*

* *be prepared for the presentation*
* *make sure your demo works on all machines and pushed to the repository*
* *make use of your time*

# SPRINT RETROSPECTIVE

## GOOD, BAD, IMPROVE, START

*In this meeting, we talk about*

* ***good*** *things that we did and happened in the sprint related to work, the team and the company*
* ***bad*** *things*
* *things to* ***improve*** *or continue*
* *things we should* ***start*** *doing which we haven't practiced yet*

# WHEN SPRINT ENDS, WE DRAW A BURNDOWN CHART THAT SHOWS HOW MANY TICKETS WE WERE ABLE TO COMPLETE IN THAT SPRINT

- **burn-down chart is great** because the team achieved all planned and committed tickets/features.
- **burn-up chart is the worst** because the team didn't accomplished what they committed and even added more commitment.
- **flat chart is not good** because the team is still working on the committed tasks/tickets.

# RETROSPECTIVE IN PRACTICE

Always Take Action Points in Sprint Retrospective and Assign Responsibility and Monitor in the Next Meeting

*Action Points are in the Next slide*

# SAMPLE ACTION POINTS FROM A REAL RETROSPECTIVE SESSION

- Start using Scrum @T
- organise knowledge sharing sessions ***@M and @J***
- organise team event ***@T***
- organise similar trainings by external trainers/consultants ***@M***
- All projects intro and status presentation ***@M***
- Find ways to develop technical skill of PO and Testers ***@M***
- Get/buy white board (ask and make sure it is bought by the company) ***@B***
- Use Slack for corporate communication ***@B***

# PAIR PROGRAMMING

# PAIR PROGRAMMING

## Advantages of Pair Programming

- **Code Quality** Pairs of developers are more likely to catch problems with code quality while working together. While the driver is focused on writing the code, the navigator is free to think about the style and function. Code quality standards tend to be followed more consistently when developers work together in pairs.
- **Robust Solutions** There are many ways to solve problems with software. Developing software involves selecting the best data structures, algorithms, and class hierarchies, among many other decisions. When developers pair, each of these choices can benefit from the experience of both developers, and the dialog they have with each other while making those choices.
- **Developer Communications** Solo development may be great for focus, but keeping developers informed of each other's work is a chronic problem. Code reviews can alleviate the problem, but pairing prevents it outright. Developers that pair are equally familiar with the code that they write, and the decisions that went into it. Teams that swap pairing partners regularly maintain a high degree of communications between all developers on a team.
- **Knowledge Transfer** Pair programming can be an excellent way for junior programmers to learn rapidly from more senior developers. Juniors and interns experience a sort of "full-immersion" in software development when pairing. They learn the entire complex process, not just by watching, but by participating.
- **Culture** The "high-bandwidth" communication of pair programming often leads to a team with great personal chemistry and a distinct team identity. Positive cultural experiences are correlated with higher rates of employee satisfaction. This is a distinct advantage in an industry where talent is expensive and scarce.

*https://www.newyorker.com/magazine/2018/12/10/the-friendship-that-made-google-huge*

*Read more on*

*https://raygun.com/blog/pair-programming-benefits/*

*https://martinfowler.com/articles/on-pair-programming.html*

*https://www.agilealliance.org/glossary/pairing*

# REFERENCES


- "Agile Software Development, Principles, Patterns, and Practices" By Robert C. Martin

  http://www.amazon.com/gp/product/0135974445

- Hands-On Full Stack Development with Spring Boot 2 and React - Second Edition

  https://www.packtpub.com/product/hands-on-full-stack-development-with-spring-boot-2-and-react-second-edition/9781838822361

- "Agile Estimating and Planning" by Mike Cohn http://www.amazon.com/gp/product/0131479415
- Scrum and XP from the trenches https://www.crisp.se/bocker-och-produkter/scrum-and-xp-from-the-trenches
- Kanban Scrum Minibook http://www.infoq.com/minibooks/kanban-scrum-minibook

- **Further Links**
  - Top 20 Agile Development Books Ever http://noop.nl/2008/06/top-20-best-agile-development-books-ever.html

# CODING TIME

*Let's get practical!*

# REQUIRED RESOURCES

- Every trainee should have a computer (preferably a high performance laptop) with high speed internet connection to install the required application packages. These are:
  - **Git version control system** https://git-scm.com/download/win
  - **Spring Boot - Backend API Development**
    - **HINT:** Installing Ubuntu on Windows will make it easy for you to install the other apps.
      - https://ubuntu.com/tutorials/ubuntu-on-windows#1-overview
    - Installing SDKMAN! **https://sdkman.io/install**
    - Spring Boot with **sdk install springboot 2.4.3**
    - Java version 11 with **sdk install java 11.0.3.hs-adpt**
    - IntelliJ IDEA Ultimate https://www.jetbrains.com/idea/download/#section=windows
    - PostgreSQL database https://www.postgresql.org/download/
    - Download Postman - a REST API Client (tester) **https://www.postman.com/downloads/**
  - **React JS - Frontend Web Development**
    - Downloads Node.js Latest LTS Version: 14.16.0 (includes npm 6.14.11) https://nodejs.org/en/download/
    - Visual Studio Code https://code.visualstudio.com/
  - **Flutter - Mobile App Development**
    - Flutter SDK https://flutter.dev/docs/get-started/install/windows
    - Download and install Android Studio https://developer.android.com/studio
    - Set up the Android emulator https://flutter.dev/docs/get-started/install/windows#set-up-the-android-emulator

# PROJECT SOURCE CODE IS AVAILABLE AT

# HTTPS://GITHUB.COM/PRACTICALSOFTWAREENGINEERINGWORKSHOP

# SDKMAN

- SDKMAN helps you to install different languages and frameworks such as Java and Spring Boot.
- You can change the java version you are using as simple as:

```
❯ sdk current java

Using java version 11.0.5-zulu

~ ..................................................
❯ sdk default java 8.0.181-zulu

Default java version set to 8.0.181-zulu

~ ..................................................
❯ sdk current java

Using java version 8.0.181-zulu
```

# BUILD AN API USING

## SPRING BOOT FRAMEWORK AND JAVA PROGRAMMING LANGUAGE

## IDE: INTELLIJ IDEA FROM JET BRAINS

# OPEN INTELLIJ IDEA AND CLICK ON 'CREATE NEW PROJECT'

# SELECT THE TYPE OF PROJECT

# INITIALISE THE PROJECT

# SELECT DEPENDENCIES

# EXERCISE

CREATE AN ACCOUNT ON GITHUB.COM
AND CREATE A REPOSITORY
AND PUSH THE CODE

# CREATE A REPOSITORY ON GITHUB.COM

# STEP 1: CREATE A REPOSITORY AND FILL INFORMATION

# STEP 2: REPOSITORY CREATED AND TIME TO PUSH YOUR CODE

Search or jump to... / Pull requests Issues Marketplace Explore

PracticalSoftwareEngineeringWorkshop / money-transfer-api

Unwatch

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

**Give access to the people you work with**

You should give access to the collaborators and teams you need to work with.

Add teams and collaborators

**Quick setup — if you've done this kind of thing before**

Set up in Desktop or HTTPS SSH git@github.com:PracticalSoftwareEngineeringWorkshop/money-transfer-api.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

**...or create a new repository on the command line**

```
echo "# money-transfer-api" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:PracticalSoftwareEngineeringWorkshop/money-transfer-api.git
git push -u origin main
```

**...or push an existing repository from the command line**

```
git remote add origin git@github.com:PracticalSoftwareEngineeringWorkshop/money-transfer-api.git
git branch -M main
git push -u origin main
```

**...or import code from another repository**

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

# GIT – VERSION CONTROL SYSTEM OR SOURCE CODE MANAGEMENT SYSTEM

› **git init**

*Initialized empty Git repository in /Users/biniam.asnake/github/moneytransfer/.git/*

› **git status**

*On branch master*

*No commits yet*

*Untracked files:*

*(use "git add <file>..." to include in what will be committed)*

*.gitignore*

*.mvn/*

*mvnw*

*mvnw.cmd*

*pom.xml*

*src/*

*nothing added to commit but untracked files present (use "git add" to track)*

# GIT – VERSION CONTROL SYSTEM OR SOURCE CODE MANAGEMENT SYSTEM

› **git add .**

› **git commit -m "Initial commit - Spring Boot project created with dependencies"**

*[master (root-commit) 95d2c1c] Initial commit - Spring Boot project created with dependencies*

*10 files changed, 758 insertions(+)*

*create mode 100644 .gitignore*

*create mode 100644 .mvn/wrapper/MavenWrapperDownloader.java*

*create mode 100644 .mvn/wrapper/maven-wrapper.jar*

*create mode 100644 .mvn/wrapper/maven-wrapper.properties*

*create mode 100755 mvnw*

*create mode 100644 mvnw.cmd*

*create mode 100644 pom.xml*

*create mode 100644 src/main/java/com/psew/moneytransfer/MoneytransferApplication.java*

*create mode 100644 src/main/resources/application.properties*

*create mode 100644 src/test/java/com/psew/moneytransfer/MoneytransferApplicationTests.java*

# GIT – VERSION CONTROL SYSTEM OR SOURCE CODE MANAGEMENT SYSTEM

› git remote add origin git@github.com:PracticalSoftwareEngineeringWorkshop/money-transfer-api.git

› git branch -M main

› git push -u origin main

*Enter passphrase for key '/Users/biniam.asnake/.ssh/id_rsa': *********

Counting objects: 26, done.

Delta compression using up to 8 threads.

Compressing objects: 100% (17/17), done.

Writing objects: 100% (26/26), 52.66 KiB | 13.16 MiB/s, done.

Total 26 (delta 0), reused 0 (delta 0)

To github.com:PracticalSoftwareEngineeringWorkshop/money-transfer-api.git

* [new branch]  main -> main

Branch 'main' set up to track remote branch 'main' from 'origin' by rebasing.*

# ! YOUR CODE IS NOW ON VERSION CONTROL SYSTEM !

# GIT CLONE – TO GET THE CODE, ANOTHER PERSON WILL CLONE THE REPOSITORY

# SETUP A POSTGRES DATABASE

# DB SCRIPT FOR CREATING DATABASE IN TERMINAL/CMD

- Connect to the database using user name 'Postgres'
  - `psql -h localhost -Upostgres`
- Create the database
  - `CREATE DATABASE moneytransfer;`
    - Expected Response: `CREATE DATABASE`
- Create a user or role
  - `CREATE ROLE moneytransferuser WITH ENCRYPTED PASSWORD '2didqGzlD6u/G1gf';`
    - Expected Response: `CREATE ROLE`
- Grant permissions/privileges to the user on the database
  - `GRANT ALL PRIVILEGES ON DATABASE moneytransfer TO moneytransferuser;`
    - Expected Response: `GRANT`
- Grant permissions to the user to be able to login and create tables
  - `ALTER USER moneytransferuser WITH LOGIN;`
    - Expected Response: `ALTER`

# CREATING A PACKAGE ON INTELLIJ IDEA

# PACKAGES

money-transfer-api
src
main
java
com
psew
moneytransferapi
Project
README.md
1: Project
7: Structure
Commit
Learn
money-transfer-api ~/github/money-transfer-api
.idea
.mvn
src
main
java
com.psew.moneytransferapi
controllers
domains
exceptions
repositories
services
wrappers
MoneyTransferApiApplication
resources
application.yml
application-prod.yml
test
java
com.psew.moneytransferapi
resources

# DEPENDENCIES IN POM.XML

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</
artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-actuator</
artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <scope>runtime</scope>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</
artifactId>
    </dependency>
    <!-- Added for validating domain classes -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-validation</
artifactId>
    </dependency>
    <dependency>
        <groupId>org.postgresql</groupId>
        <artifactId>postgresql</artifactId>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</
artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
```

# ACCOUNT DOMAIN/ENTITY

Account.java

```java
package et.edu.uog.moneytransferapiuog.domains;

import lombok.Data;

import javax.persistence.*;

/**
 * @author Biniam Asnake
 */
@Data
@Entity
public class Account {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false)
    private String firstName;

    @Column(nullable = false)
    private String lastName;

    @Column(nullable = false)
    private String phoneNumber;

    @Column(nullable = false)
    private String email;

    @Column(nullable = false)
    private Integer pin;

    private Double balance = 0.0;

    private Boolean isVerified = true;
}
```

# DATABASE CONFIGURATION

# APP – RUNNING AND TABLE CREATED

# APP HEALTH STATUS – POSSIBLE BY SPRING ACTUATOR DEPENDENCY

localhost:8080/actuator/health


```
{
    "status": "UP"
}
```

# BASICS OF RESTFUL API

# BASICS OF RESTFUL API

- Named Resources/Endpoints
  - E.g.
    - /api/account/create
    - /api/account/list
- HTTP Methods
  - GET - to read data
  - POST - to insert/create data
  - PUT - to update data
  - DELETE -  to delete data

# INTERNAL COMMUNICATION PATTERN

Author Biniam Asnake

# ACCOUNT CONTROLLER, SERVICE AND REPOSITORY

money-transfer-api-act › src › main › java › com › act › moneytransferapi › respcsitories › AccountRepository

AccountController.java

```java
/** @author Biniam Asnake */
@RestController
@RequestMapping("/api/account")
public class AccountController {

    @Autowired
    private AccountService accountService;

    @PostMapping(value = "/create", produces = "application/json")
    public Account createAccount(@RequestBody Account accountRequest) {
        return accountService.createAccount(accountRequest);
    }
```

AccountService.java

```java
@Service
public class AccountService {

    @Autowired
    private AccountRepository accountRepository;

    public Account createAccount(Account account) {
        return accountRepository.save(account);
    }
```

AccountRepository.java

```java
/** @author Biniam Asnake */
@Repository
public interface AccountRepository extends CrudRepository<Account, Long> {}
```

# CREATING DB CONFIG

# DB CONFIG IN INTELLIJ

# DB STRUCTURE IN INTELLIJ

# ACCOUNT CONTROLLER INTEGRATION TEST

```java
@RunWith(SpringRunner.class)
@SpringBootTest
@AutoConfigureMockMvc
@AutoConfigureTestDatabase
@ActiveProfiles("test")
public class AccountControllerIntegrationTest {
    @Autowired private MockMvc mockMvc;
    @Autowired protected ObjectMapper objectMapper;
    @Test
    public void shouldCreateAccountGivenCorrectInput() throws Exception {
        mockMvc.perform(
                post( urlTemplate: "/api/account/create")
                        .accept("application/json")
                        .contentType("application/json")
                        .content("{\n" +
                                "    \"firstName\" : \"Biniam\",\n" +
                                "    \"lastName\" : \"Asnake\",\n" +
                                "    \"email\": \"biniamasnake@gmail.com\",\n" +
                                "    \"phoneNumber\": \"+251 966-272502\",\n" +
                                "    \"pin\": \"1234\"\n" +
                                "}"))
                .andDo(print())
                .andExpect(status().isOk())
                .andExpect(content().json(
                        objectMapper.writerWithDefaultPrettyPrinter().writeValueAsString(
                        objectMapper.readValue(new File(
                                pathname: "src/test/json/create_account_response.json"),
                                Object.class))));
    }
```

## Another Test

src/test/java/com/psew/moneytransferapi/AccountControllerIntegrationTest.java

```
@RunWith(SpringRunner.class)
@SpringBootTest
@AutoConfigureMockMvc
@AutoConfigureTestDatabase
@ActiveProfiles("test")

public class AccountControllerIntegrationTest {

    @Autowired private MockMvc mockMvc;

    @Autowired protected ObjectMapper objectMapper;

    @Test
    public void shouldCreateAccountGivenCorrectInput() throws Exception {
        mockMvc.perform(
                post("/api/account/create")
                        .accept("application/json")
                        .contentType("application/json")
                        .content("{\n" +
                                "    \"firstName\" : \"Biniam\",\n" +
                                "    \"lastName\" : \"Asnake\",\n" +
                                "    \"email\": \"biniamasnake@gmail.com\",\n" +
                                "    \"phoneNumber\": \"+251 966-272502\",\n" +
                                "    \"pin\": \"1234\"\n" +
                                "}"))
                .andDo(print())
                .andExpect(status().isOk())
                .andExpect(content().json(
                        objectMapper.writerWithDefaultPrettyPrinter().writeValueAsString(
                        objectMapper.readValue(new File(
                                "src/test/json/create_account_response.json"),Object.class))));
    }
}
```

## src/test/java/com/psew/moneytransferapi/AccountControllerIntegrationTest.java

```java
@RunWith(SpringRunner.class)
@SpringBootTest
@AutoConfigureMockMvc
@AutoConfigureTestDatabase
public class AccountControllerIntegrationTest {

    @Autowired
    MockMvc mvc;

    @Test
    public void shouldReturnAnErrorWhenCreatingAccountWithDuplicateData() throws Exception {

      String data = "{" +
              "  \"firstName\": \"Biniam\"," +
              "  \"lastName\": \"Asnake\"," +
              "  \"email\" : \"biniamasnake@gmail.com\"," +
              "}";

      mvc.perform(
              post("/api/account/create")
                      .accept(MediaType.APPLICATION_JSON)
                      .contentType(MediaType.APPLICATION_JSON)
                      // Here we are using the Account object defined in the above 'data' JSON
                      .content(data)
      )
              .andDo(print())
              .andExpect(status().isBadRequest());
    }
  }
```

# UNIT TEST FOR A SERVICE CLASS USING JUNIT AND MOCKITO

```java
@RunWith(MockitoJUnitRunner.class)
public class UserServiceTests {

    @Mock
    UserRepository userRepository;

    @InjectMocks
    UserService userService;

    @Test
    public void when_save_user_it_should_return_user() {
        CreateUserRequest createUserRequest = new CreateUserRequest();
        createUserRequest.setName("test user");

        when(userRepository.save(any(User.class))).thenReturn(new User());

        User created = userService.save(createUserRequest);

        assertThat(created.getName()).isSameAs(createUserRequest.getName());
    }
}
```

# CI/CD PIPELINE

# THE TESTING AND DEPLOYMENT PIPELINE WE WILL USE

Code → Push to GitHub → Run tests on Travis CI → Deploy code to production using Heroku

# TRAVIS CONFIGURATION

money-transfer-api 〉 .travis.yml

Project

- money-transfer-api ~/github/money-transfer-api
  - .idea
  - .mvn
  - src
  - target
  - testing-endpoints
  - .gitignore
  - .travis.yml
  - HELP.md
  - money-transfer-api.iml
  - mvnw
  - mvnw.cmd
  - pom.xml
  - Procfile
  - README.md
  - system.properties
- External Libraries
- Scratches and Consoles

.travis.yml

```yaml
language: java
jdk:
- oraclejdk11
script:
- "./mvnw clean verify"
branches:
  only:
  - main
deploy:
  provider: heroku
  api_key: $HEROKU_API_KEY
  app: money-transfer-api-biniam
  on:
    repo: PracticalSoftwareEngineeringWorkshop/money-transfer-api
    branch: main
  skip_cleanup: 'true'
```

*Checkout the next slide on how to configure the API key*

# ENABLING CROSS ORIGIN REQUESTS TO ENABLE THE WEB APP TO SEND API REQUEST

```
package com.psew.moneytransferapi;

@SpringBootApplication
public class MoneyTransferApiApplication {

    public static void main(String[] args) {
        SpringApplication.run(MoneyTransferApiApplication.class, args);
    }

    @Bean
    public WebMvcConfigurer corsConfigurer() {
        return new WebMvcConfigurer() {
            @Override
            public void addCorsMappings(CorsRegistry registry) {
                registry.addMapping("/**")
                        .allowedOrigins("*");
            }
        };
    }
}
```

Read more on https://spring.io/guides/gs/rest-service-cors/

# ADDING HEROKU API KEY TO TRAVIS CI

Click on account settings on Heroku and copy the api key

# ADDING HEROKU API KEY TO TRAVIS CI

Click on **More Options** on Travis CI and the api key as Environment Variable

# TRAVIS CI – CONTINUOUS INTEGRATION AND DEPLOYMENT

# PROCFILE FOR RUNNING THE APP IN HEROKU

```
web: java -Dserver.port=$PORT
-Dspring.profiles.active=production $JAVA_OPTS -jar target/
money-transfer-api-1.0.0-SNAPSHOT.jar
```

# SYSTEM.PROPERTIES FILE FOR TELLING HEROKU WE ARE USING JAVA 11

```
java.runtime.version = 11
```

# ADDING A DATABASE TO HEROKU – CLICK ON 'CONFIGURE ADDONS' AND SEARCH FOR 'POSTGRES'

*Get the credentials and it to src/application-production.yml file*

# HEROKU – DEPLOYING THE APP

# HEROKU – DEPLOYED API SHOWS THIS RESULT

*https://money-transfer-api-biniam.herokuapp.com/api/account/list*

money-transfer-api-biniam.herokuapp.com


```
[
  - {
        "id" : 1,
        "firstName" : "Biniam",
        "lastName" : "Asnake",
        "email" : "biniamasnake@gmail.com",
        "phoneNumber" : "+251966272502",
        "pin" : 1234,
        "balance" : 0,
        "isVerified" : true,
    },
  + {...} (8 hidden elements),
  + {...} (8 hidden elements),
  - {
        "id" : 6,
        "firstName" : "Abreham",
        "lastName" : "Gebremedin",
        "email" : "abrehamgebremedin12@gmail.com",
        "phoneNumber" : "33333333333333333333",
        "pin" : 33333,
        "balance" : 0,
        "isVerified" : true,
    },
  + {...} (8 hidden elements),
  + {...} (8 hidden elements),
  + {...} (8 hidden elements),
  + {...} (8 hidden elements),
  + {...} (8 hidden elements),
  + {...} (8 hidden elements),
  + {...} (8 hidden elements),
  + {...} (8 hidden elements)
]
```

# Link to Code

**https://github.com/PracticalSoftwareEngineeringWorkshop/money-transfer-api**

# DEVELOPING THE WEB APP

# BASICS OF REACT JS

- Create a react component

```
import React from 'react';
const CreateAccount = () => {
    // component code like states, side effects and functions goes here
    return (
        // UI code goes here
    );
}
```

- Create a state variable to store data

```
const [isSubmitButtonClicked, setIsSubmitButtonClicked] = useState(false);
```

- Create an effect to run when a state/variable changes

```
useEffect(() => {
    // function code goes here
}, [params]);
```

Learn about **React Hooks** at https://reactjs.org/docs/hooks-intro.html and https://reactjs.org/docs/hooks-overview.html

There are many **React UI Component Frameworks** but the common one is Material UI (www.material-ui.com)

More on: https://www.codeinwp.com/blog/react-ui-component-libraries-frameworks/

# SEARCHING FOR COMPONENTS IN MATERIAL UI TO SEE EXAMPLE AND DOCUMENTATION

# REACT JS – INITIAL PROJECT SETUP

# REACT JS – INITIAL APPLICATION CLEANED UP – UNNECESSARY FILES REMOVED

# REACTJS – CREATE ACCOUNT COMPONENT

CreateAccount.js M

src > CreateAccount.js > CreateAccount

```
1  import React, { useState, useEffect } from 'react';
2  import { TextField, Button } from '@material-ui/core';
3  import axios from 'axios';
4
5  const CreateAccount = () => {
6      const [firstName, setFirstName] = useState("");
7      const [isSubmitButtonClicked, setIsSubmitButtonClicked] = useState(false);
8
9      useEffect(() => {
10         if (isSubmitButtonClicked === true) {
11             // make api call
12             axios.post("http://localhost:8080/api/account/create", {
13                 "firstName": firstName
14             }
15             )
16                 .then((response) => alert("Account created with id " + response.data.id))
17                 .catch((error) => console.error(error));
18         }
19     }, [isSubmitButtonClicked, firstName]);
20
21     return (
22         <form>
23             <TextField
24                 id="firstName"
25                 label="First Name"
26                 variant="outlined"
27                 autoFocus
28                 value={firstName}
29                 onChange={(event) => { setFirstName(event.target.value) }}
30             />
31             <Button
32                 variant="contained"
33                 color="primary"
34                 size="small"
35                 onClick={() => setIsSubmitButtonClicked(true)}
36             >
37                 Save
38         </Button>
39         </form>
40     );
41 }
42 export default CreateAccount;
```

main* ⊗ 0 ⚠ 0 {..} : Scanning.. Ln 23, Col 23 Spaces: 4 UTF-8 LF JavaScript ESLint

Money Transfer API - ACT

localhost:3000/create

Money Transfer App CREATE ACCOUNT LIST ACCOUNT

First Name SAVE

# REACTJS – LIST ACCOUNTS COMPONENT

CreateAccount.js M | ListAccounts.js U | App.js 1, M

src > ListAccounts.js > ListAccounts

```
import React from 'react';
import { API_URL } from './Constants';
import useAxios from 'axios-hooks';
import { TableContainer, Table, TableHead, TableBody, TableRow, TableCell

const ListAccounts = () => {

    const [{ data, loading, error }] = useAxios(`${API_URL}/api/account/l
    if (loading === true) return ("Loading ...")
    if (error) return ("Error happened!")
    return (
        <TableContainer>
            <Table>
                <TableHead>
                    <TableRow>
                        <TableCell>ID</TableCell>
                    </TableRow>
                </TableHead>
                <TableBody>
                    {data.map((row) => {
                        return (
                            <><TableRow>
                                <TableCell>{row.id}</TableCell>
                            </TableRow></>
                        )
                    }
                    )}
                </TableBody>
            </Table>
        </TableContainer>
```

main*+ | 0 1 | {..}: Scanning.. | UTF-8 | LF | JavaScript | ESLint

localhost:3000

Money Transfer App CREATE ACCOUNT

| ID |
|---|
| 1 |
| 2 |
| 3 |
| 5 |
| 15 |
| 17 |
| 18 |
| 21 |
| 22 |
| 23 |
| 25 |
| 26 |
| 27 |

# DEVELOPING THE MOBILE APP

# CREATING AND RUNNING A FLUTTER APP

```
flutter create money_transfer_app
cd money_transfer_app
flutter run

# To run on all emulators at once
flutter run -d all
```

# IMPLEMENTING SHOW APP SCREEN ON FLUTTER UI

```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Money Transfer App',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(primarySwatch: Colors.blue,),
      home: HomePage(title: 'Money Transfer'),
    );
  }
}
```

# ... IMPLEMENTING SHOW APP SCREEN ON FLUTTER UI

```
import 'package:flutter/material.dart';
import 'dart:convert' as convert;
import 'package:http/http.dart' as http;

class HomePage extends StatefulWidget {
  HomePage({Key key, this.title}) :
super(key: key);

  final String title;

  @override
  _HomePageState createState() =>
_HomePageState();
}

class _HomePageState extends State<HomePage>
{
  var id, firstName, lastName;

  void apiCall() async {
    var url = Uri.https(
        "money-transfer-api-
act.herokuapp.com", "/api/account/show/1");

    var response = await http.get(url);

    var jsonResponse =
convert.jsonDecode(response.body);

    print(jsonResponse);

    setState(() {
      id = jsonResponse['id'];
      firstName = jsonResponse['firstName'];
      lastName = jsonResponse['lastName'];
    });
  }

  @override
  void initState() {
    super.initState();
    apiCall();
  }
```

# ... IMPLEMENTING SHOW APP SCREEN ON FLUTTER UI

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(title: Text(widget.title),),
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          Text('Show Account',),
          Text('First Name $firstName’, style: TextStyle(fontSize: 30),),
          Text('Last Name $lastName’, style: TextStyle(fontSize: 30),
          ),
        ],
      ),
    ),
  );
}
```

# RESULT OF IMPLEMENTING SHOW APP SCREEN ON FLUTTER UI

*Thank you for attending the workshop.*
*I hope you learned how to develop a real life application from idea to releasing.*

*Please give Feedback about the workshop using the following form*

# HTTP://BIT.LY/AGILE-WORKSHOP-FEEDBACK