Modalitäten: Eigenleistung in der Gruppe

Ostbayerische Technische Hochschule Amberg-Weiden

- **Die Einzelanteile müssen ersichtlich sein!**
  - Gehen Sie davon aus, dass eine **git log Analyse** stattfindet.
    - Ein solches Werkzeug ist bspw. https://github.com/ejwa/gitinspector
  - Eigene Programmieranteile sind **unter eigenem Namen zu commiten**!
    - Die folgenden Einstellungen sind Pflicht und **notenrelevant**!

```
git config --global user.name "Musterfrau, Erika"
git config --global user.email e.musterfrau@oth-aw.de
```

  - (Vorsicht vor git squash! In der industriellen Praxis sind kompakte Historien wertvoll. Dennoch: In unserem Umfeld ist squash nicht empfohlen.)
- **Benotungsrelevant sind nur Commits auf dem Branch „master“.**
  - Überlegen Sie sich eine git Vorgehensweise, um Änderungen erst auf Feature-Branches zu entwickeln und gemeinsam zu integrieren
    - vgl. Vorgehensweise **gitflow** o.ä. → Sie können zunächst auch auf einem Branch wie „develop“ Ihre Feature-Branches integrieren, aber…
  - Vergessen Sie **vor dem PrA-Review** Termin nicht einen finalen git merge auf den **master**-Branch

Modalitäten: Zusammenarbeit

Ostbayerische Technische Hochschule Amberg-Weiden

- **Peer Programming ist legitim: Sie müssen lediglich regelmäßig und mit alternierendem Namen committen.**
  - Ähnlich: Mob Programming! (Schauen Sie ggf. unter diesem Stichwort im Netz nach Tool-Unterstützung, git Vorgehensweisen und Best Practices rund um Mob Programming)
  - Konsequenz: Sie verantworten dann jeder einzeln letztlich alle derart programmierten Artefakte. Das ist sehr gut!

- **Keine Aufteilung "Ich mach die Präsentationen, du programmierst".**
  - Eine **Aufgabenteilung** nach fachlichen Modulen ist völlig i.O.
  - Eine Aufgabenteilung nach technischen Themenkomplexen ist i.O.
  - Eine Aufgabenteilung nach technischen Schichten ist fragwürdig und nicht zeitgemäß
  - Eine Aufgabenteilung nach User Story ist fragwürdig und nicht zeitgemäß
    - Im Zweifelsfall ist das ganze Team für jede einzelne User Story gemeinschaftlich verantwortlich. Helfen Sie sich gegenseitig!

Exkurs: gitinspector

Ostbayerische Technische Hochschule Amberg-Weiden

Identisch mit „Nachname 4, Vorname 4“:
Initiales erstellen des Repos inkl. MarkDown-Datei per GitLab Web-UI und dort keinen Klarnamen hinterlegt.
(x5m7 ist seine technische GitLab-Kennung)

Author	Commits	Insertions	Deletions	% of changes
Nachname1, Vorname1	31	787	330	17.52
x5m7	1	2	0	0.03
Nachname2, Vorname2	7	28	17	0.71
Nachname3, Vorname3	49	1110	361	23.08
Nachname4, Vorname4	17	1510	309	28.54
c9c0	21	1523	397	30.12

Trittbrettfahrer?
(ggf. Nicht-Bestanden)

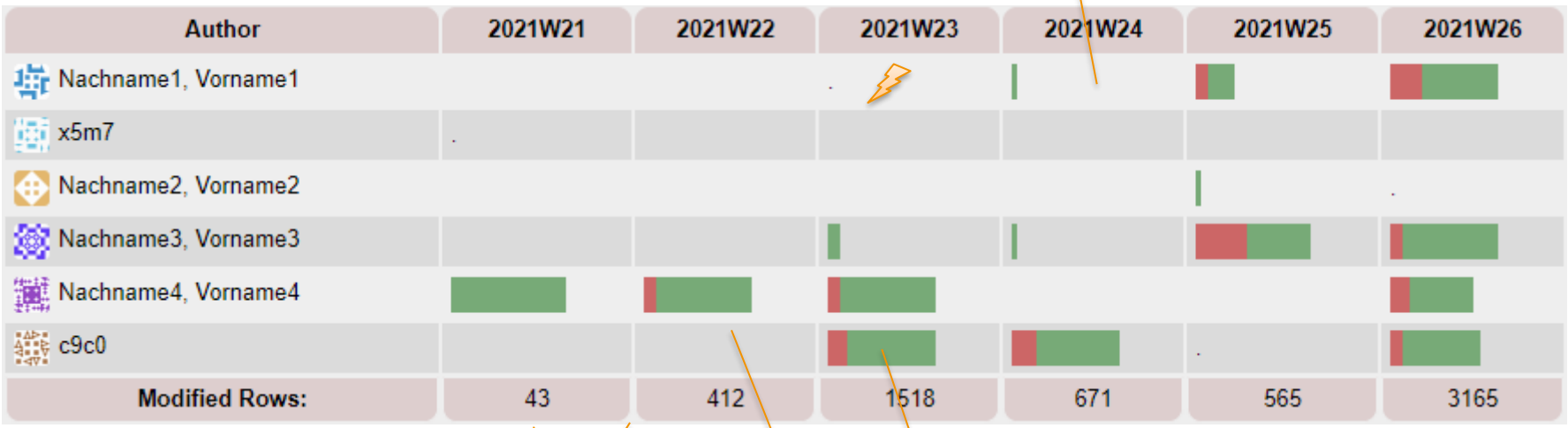
5. Teammitglied: Hat `git config` vergessen ☹
(c9c0 ist seine technische GitLab-Kennung)

Denken Sie (**von Anfang an!**) an **`git config`!**

Exkurs: gitinspector

Ostbayerische Technische Hochschule Amberg-Weiden

Person1 erbringt erst in den letzten beiden Wochen signifikante Beiträge. ☹



Author	2021W21	2021W22	2021W23	2021W24	2021W25	2021W26
Nachname1, Vorname1						
x5m7						
Nachname2, Vorname2						
Nachname3, Vorname3						
Nachname4, Vorname4						
c9c0						
Modified Rows:	43	412	1518	671	565	3165

Vorarbeiten

Offizieller Beginn

Gruppe orientiert sich gemeinsam

Allerspätestens in der zweiten Woche Commiten „Sehr Gut“-Studierende!

Exkurs: gitinspector